الكلية التطبيقية

قسم تقنيات الحاسوب

السنة الرابعة

مقرر البرمجيات النقالة

مدرس المقرر : م. رود الأصفر

المرجع: كتاب أساسيات برمجة تطبيقات الهواتف الذكية باستخدام نظام أندرويد للدكتور إياد محمد قاسم الأغا

التراجم

ترجمة المصطلحات التقنية المتعلقة بالبرمجة الشيئية ونظام أندرويد كان من الصعوبات التي واجهت المؤلف في إخراج الكتاب. لشرح المصطلحات التقنية باللغة العربية بدون الإخلال بمفاهيم هذه المصطلحات قمنا بترجمة المصطلحات إلى العربية وقمنا أثناء الشرح بذكر الترجمة العربية ملحوقة بالمصطلح الإنجليزي بين قوسين. التراجم المستخدمة للمصطلحات المختلفة موضحة في الجدول التالي. نود التأكيد على أن بعض المصطلحات لم تترجم حرفياً حيث تم الاعتماد أحياناً على تراجم قريبة لكونها أقرب للمفهوم الفعلي من الترجمة الحرفية من وجهة نظر المؤلف.

من الضروري أيضاً أن نؤكد على أن ترجمة المصطلحات هي بغرض مساعدة القارئ العربي على فهم السياق وليس لاستبدال المصطلحات الأصلية بالتراجم العربية. لذلك نوصي كلاً من المدرس والطالب بالحرص على استعمال المصطلح الإنجليزي قدر الإمكان.

المصطلح	الترجمة المستخدمة في الكتاب
Activity	الفعالية
Service	الخدمة
Broadcast Receivent	مستقبل النشر
Manifest File	ملف الوثيقة
Layout File	ملف التصميم
Assets Folder	مجلد الممتلكات
Activity Lifecycle	دورة حياة الفعالية
Backstack	الحافظة الخلفية
Intent	الهدف
PendingIntent	الهدف المعلق
Explicit Intent	الهدف الصريح
Implicit Intent	الهدف المضمن
Intent Filter	مرشح الهدف
Action	حدث أو إجراء
Class	فئة
Subclass	فئة فرعية
Superclass	فئة رئيسية
Interface	الواجهة البرمجية
Object	كائن
Method	دالة
Callback Method	دالة الاتصال الراجع
Constructor	الباني
Package	حزمة
Listener	مستمع
Adapter	محول

ListView	قائمة العرض
Shared Preferences	التفضيلات المشتركة
Bundle	الحزمة
OptionsMenu	قائمة الخيارات
Context Menu	قائمة السياق
Popup Menu	القائمة المنبثقة
MenuItem	عنصر القائمة
Action Bar	شريط العمل
Action Overflow	ملحق شريط العمل
Navigation Drawer	درج التصفح
XML Attribute	خاصية XML
Permission	إذن
Notification	إشعار
Notification Drawer	درج الإشعارات
Content Provider	مزود المحتوى
Content Resolver	مستخرج المحتوى
Fragment	القطعة
Fragment Manager	مدير القطع
Dialog	مربع الحوار

الوحدة الأولى:

الفعالية (Activity)

يتعلم الطالب في هذه الوحدة:

- ✓ . مفهوم الفعالية (Activity) في نظام أندرويد وطريقة إنشائها.
- ✓ . دورة حياة الفعالية (Activity) وطريقة إدارتها بصورة سليمة.
- ✓ . حفظ حالة الفعالية (Activity) واستردادها.
- ✓ التعريف بأوضاع العمل الخاصة بالفعالية (Activity) واستخدام الحافظة الخلفية (Back stack).
- ✓ . التدريب العملي على إنشاء الفعالية (Activity) ومعالجة حالاتها المختلفة.

هناك أنواع مختلفة من المكونات يمكن أن يبنى منها تطبيق الأندرويد. أول هذه المكونات يسمى بالفعالية (Activity) وهي جزء من تطبيق أندرويد يوفر للمستخدم واجهة تفاعلية تمكنه من تنفيذ أمر ما مثل تصفح الأخبار، البحث عن معلومة، الاتصال الهاتفي، التقاط الصور، عرض خريطة أو أي مهمة أخرى. كل فعالية (Activity) تمثل نافذة مستقلة في التطبيق. هذه النافذة قد تعرض في وضع ملء الشاشة أو قد تحتل جزء صغير من الشاشة.

تطبيق الأندرويد يتكون عادة من مجموعة من الفعاليات (Activities) الغير مرتبطة ببعضها. كل تطبيق له عادةً فعالية رئيسية واحدة (Main Activity) وهي التي تعرض على الشاشة عن تشغيل التطبيق. بعد تشغيل الفعالية الرئيسية (Main Activity) ، يمكن من خلالها تشغيل فعاليات أخرى لتنفيذ مهمات مختلفة. على سبيل المثال، عند تشغيل تطبيق إخباري لأول مرة يتم عرض قائمة الأخبار في الواجهة الخاصة بالفعالية الرئيسية (Main Activity)، وعند اختيار أي خبر محدد بالنقر عليه، يتم تشغيل فعالية جديدة لعرض تفاصيل الخبر.

واجهة المستخدم يتم عادةً إنشاؤها وتصميمها من خلال ملفات XML كما درست مسبقاً. الفعالية (Activity) تقوم عند بدء تشغيلها بإنشاء الواجهة بناءً على محتوى ملف XML. في ملف الفعالية (Activity) يتم كتابة كود يمثل الإجراءات المختلفة التي يجب تنفيذها عند تفاعل المستخدم مع عناصر الواجهة. وبذلك يكون هناك فصل كامل بين تصميم الواجهات، والذي يتم من خلال ملفات XML، والإجراءات اللازم تنفيذها، والتي يتم تحديدها داخل الفعالية (Activity).

قبل الحديث عن التعامل مع واجهة المستخدم والتفاعل مع الأحداث المختلفة UI Events، سيتم شرح كيفية إنشاء الفعالية (Activity) وإدارتها.

إنشاء الفعالية (Creating the Activity)

لإنشاء فعالية جديدة، يجب عليك إنشاء فئة فرعية (subclass) من الفئة (Activity) أو أي فئة متفرعة من Activity). بعد ذلك عليك كتابة الكود الخاص بدوال الاتصال الراجع (Callback Methods). دوال الاتصال الراجع هي دوال يتم تنفيذها تلقائيا من قبل النظام (بدون تدخل المستخدم) بناء على التغير في دورة حياة الفعالية (Activity Life Cycle).

فمثلاً عند بدء تشغيل الفعالية يقوم النظام بتشغيل الدالة ()onCreate، وعند إزالة الفعالية (Activity) يتم تشغيل الدالة ()onDestroy. يمكن للمستخدم تنفيذ أي كود عند بدء تشغيل الفعالية (Activity) عن طريقة إضافة هذا الكود إلى الدالة ()onCreate. فمثلاً، نقوم عادة بكتابة الكود الخاص بإنشاء محتويات واجهة المستخدم داخل الدالة ()onCreate وذلك لتجهيز الواجهة للعرض عن بداية تشغيل الفعالية (Activity). أهم الدوال التي نحتاج عادةً إلى إضافة كود لها عند إنشاء الفعالية (Activity) هي:

- ()onCreate: يجب كتابة الكود الخاص بهذه الدالة والتي ينفذها النظام تلقائياً عند إنشاء الفعالية. نقوم عادةً في هذه الدالة بتهيأة وانشاء المكونات المختلفة الخاصة بالفعالية. من خلال هذه الدالة يتم تحديد هيكلية الواجهة الخاصة بالفعالية عن طريق استدعاء الدالة ()setContentView. نقوم عادة بتحديد معرف خاص بملف XML الخاص بالواجهة وتمريره إلى الدالة ()setContentView.
- ()onResume: يتم تنفيذ هذه الدالة عندما تصبح الفعالية مشاهدة visible وذلك بعد إنشائها أو استئناف عملها بعد توقف. أي أن هذه الدالة يتم تنفيذها بعد الدالة ()onCreate.قد تستخدم هذه الدالة لاسترجاع حالة الفعالية أو لربطها بمكونات التطبيق الأخرى مثل الخدمات (Services).
- ()onPause: يقوم النظام باستدعاء هذه الدالة تلقائياً عند ايقاف الفعالية. إيقاف الفعالية قد يكون نتيجة تشغيل فعالية اخرى تحجبها جزئياً عن المشاهدة، ولا يعني بالضرورة إنهاء الفعالية أو إخفائها. نحتاج إلى كتابة هذه الدالة لحفظ حالة الفعالية والبيانات التي تستخدمها حتى لا تفقد، وحتى يتم استعادتها عن الرجوع للفعالية.
- ()onStop: يتم تنفيذ هذه الدالة عندما تصبح الفعالية غير مشاهدة تماماً. يمكن استخدام هذه الدالة لإيقاف العمليات التي تنفذها الفعالية.

هناك المزيد من دوال الاتصال الراجع (Callback Methods) والتي سيتم التطرق لها عند الحديث عن دورة حياة الفعالية (Activity Life Cycle) لاحقاً في هذه الوحدة.

الإعلان عن الفعالية في ملف القائمة Manifest

بعد إنشاء الفئة (class) الخاصة بالفعالية، يجب الاعلان عن الفعالية في ملف الوثيقة Manifest. ملف الوثيقة Manifest يحدد الإعدادات المختلفة للتطبيق والمكونات التي يستخدمها وسيتم شرحه في الوحدة الثانية من هذا الكتاب. الإعلان عن فعالية ما (Activity) يتم عن طريق إضافة الخاصية <activity> داخل الخاصية <application> كما هو موضح في الكود التالي (ملاحظة: قد تقوم بيئة العمل مثل Eclipse بعمل ذلك تلقائياً):

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest>
```

هناك أيضاً بعض الخصائص التي تضاف داخل المكون <activity> والخاصة بالفعالية مثل اسم الفعالية android:name والذي يجب اضافتها لتحديد اسم الفئة (class) الخاص بالفعالية.

إدارة دورة حياة الفعالية (Managing the Activity Life Cycle)

إدارة دورة حياة الفعالية (Activity) عن طريق كتابة الكود الخاص بدوال الاتصال الراجع (Callback Methods) إجراء مهم لبناء تطبيق صحيح. دورة حياة الفعالية (Activity) تتأثر مباشرةً بارتباطها بالفعاليات الأخرى. الفعالية يمكن أن تكون في حالة من ثلاثة حالات:

- وضع العمل running: وفي هذه الحالة تكون الواجهة في المقدمة ومفعلة من قبل المستخدم.
- وضع الإيقاف المؤقت paused: تنتقل فعالية ما إلى وضع paused عندما يتم تشغيل فعالية أخرى تنتقل إلى الواجهة وتكون هذه الفعالية الجديدة شفافة جزئياً أو لا تملأ الشاشة بحيث لا يزال بالإمكان رؤية الفعالية الأولى. الفعالية في وضع الإيقاف المؤقت تظل الفعالية في الذاكرة وتحتفظ بحالتها وقيم المتغيرات فيها، ولكن قد يلجأ النظام لتدميرها في حالة القصور الشديد في الذاكرة.
- وضع الإيقاف stopped: تنتقل الفعالية إلى هذه الحالة عندما تصبح غير مشاهدة تماماً ويتم حجبها خلف فعالية أخرى (أي أنها تصبح في المؤخرة). الفعالية في وضع الايقاف تحتفظ كذلك بحالتها وقيم المتغيرات فيها، ولكن قد يلجأ النظام لتدميرها في حالة القصور الشديد في الذاكرة.

عندما تنتقل الفعالية من حالة إلى أخرى من الحالات الموضحة أعلاه، يتم إعلام الفعالية من قبل النظام عن طريق تشغيل دوال الاتصال الراجع (Callback Methods) الملائمة حسب الحالة. لذلك، يمكن كتابة كود في هذه الدوال لتنفيذ مهام محددة عندما تتغير حالة الفعالية. الكود الموضح بالأسفل يوضح هيكلية الفعالية وأهم دوال الاتصال الراجع(Callback Methods):

```
public class ExampleActivity extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        // الفعالية في مرحلة الانشاء.
    }
    @Override
    protected void onStart(){
        super.onStart();
        // الفعالية على وشك أن تصبح مشاهدة. تنفذ هذه الدالة قبل العرض مباشرة.
    }
    @Override
    protected void onResume(){
        super.onResume();
        // تنفذ هذه الدالة بعد ان تصبح الدالة مشاهدة
    }
```

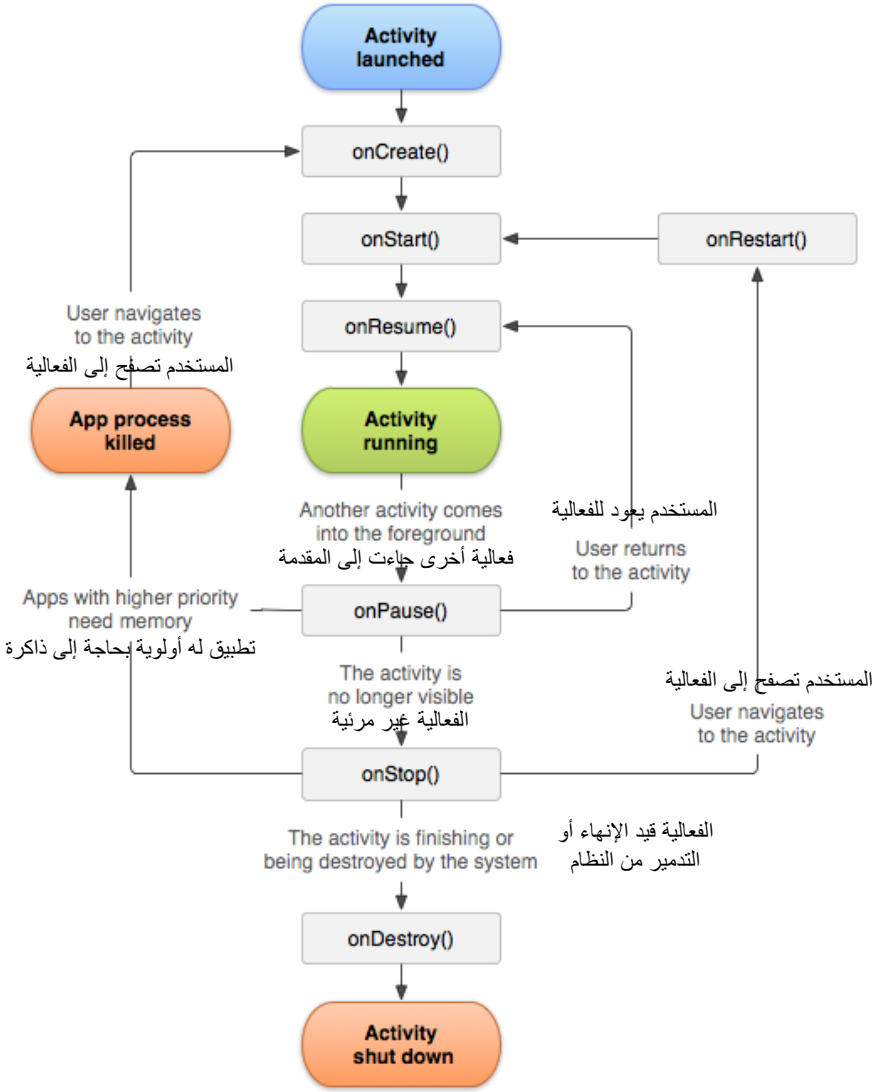
```
    @Override
    protected void onPause(){
        super.onPause();
        تنفذ هذه الدالة عن ايقاف الفعالية مؤقتا بسبب تشغيل فعالية أخرى//
    }
    @Override
    protected void onStop(){
        super.onStop();
        // الفعالية غير مشاهدة تماما وعليه تم ايقافها وتنفيذ هذه الدالة
    }
    @Override
    protected void onDestroy(){
        super.onDestroy();
        تنفذ هذه الدالة قبل تدمير الفعالية مباشرةً//
    }
}
```

ملاحظة: اذا أردت كتابة أي كود في دوال الاتصال الراجع فعليك أولاً تنفيذ دوال الاتصال الراجع في الفئة الأم (superclass) وذلك قبل تنفيذ أي كود جديد كما هو موضح بالشكل. فمثلاً، في الدالة ()onCreate يتم تنفيذ الدالة ()super.onCreate ثم يتم كتابة بقية الكود.

دورة حياة الفعالية كاملةً موضحة بالشكل 1-1 بالأسفل حيث تمثل المستطيلات دوال الاتصال الراجع (Callback Methods)، وتمثل الدوائر حالات الفعالية. دورة حياة الفعالية تحصل ما بين تنفيذ الدالة ()onCreate والدالة ()onDestroy. لذلك، كل ما يرتبط بإنشاء الفعالية من إنشاء هيكلية واجهة المستخدم والمتغيرات وتهيئة حالة الفعالية والاتصال بقواعد البيانات مثلا يتم في الدالة ()onCreate. كل ماهو مرتبط بانهاء حالة الفعالية من اغلاق الاتصال بالخدمات أو قواعد البيانات يمكن أن يتم في الدالة ()onDestroy.

هناك ما يسمى دورة الحياة المشاهدة للفعالية Visible lifecycle وتحصل ما بين الدالة ()onStart و ()onStop وخلالها تكون الفعالية مشاهدة من قبل المستخدم. لاحظ أن دورة الحياة المشاهدة هي جزء من دورة الحياة الكاملة. لاحظ أن الفعالية المشاهدة قد لا تكون بالضرورة في وضع العمل running (قد تكون في وضع الايقاف الموقت نتيجة تشغيل فعالية تحجبها جزئياً).

وهناك أيضا دورة الحياة في المقدمة Foreground lifecycle وهي جزء من الدورتين السابقتين وتحصل بين تنفيذ الدالة ()onResume والدالة ()onPause. خلال هذه الفترة تكون الفعالية في وضع العمل running، أي أنها تكون مشاهدة ويتم التفاعل معها من قبل المستخدم. لاحظ أن الكود الذي يتم كتابتة في الدالة ()onResume و الدالة ()onPause يتم تنفيذه عندما يحصل أي تغيير في حالة الفعالية، ولذلك تستخدم هذه الدوال بكثرة لحفظ حالة الفعالية والبيانات المتعلقة بها عند حصول أي تغيير في حالة الفعالية.

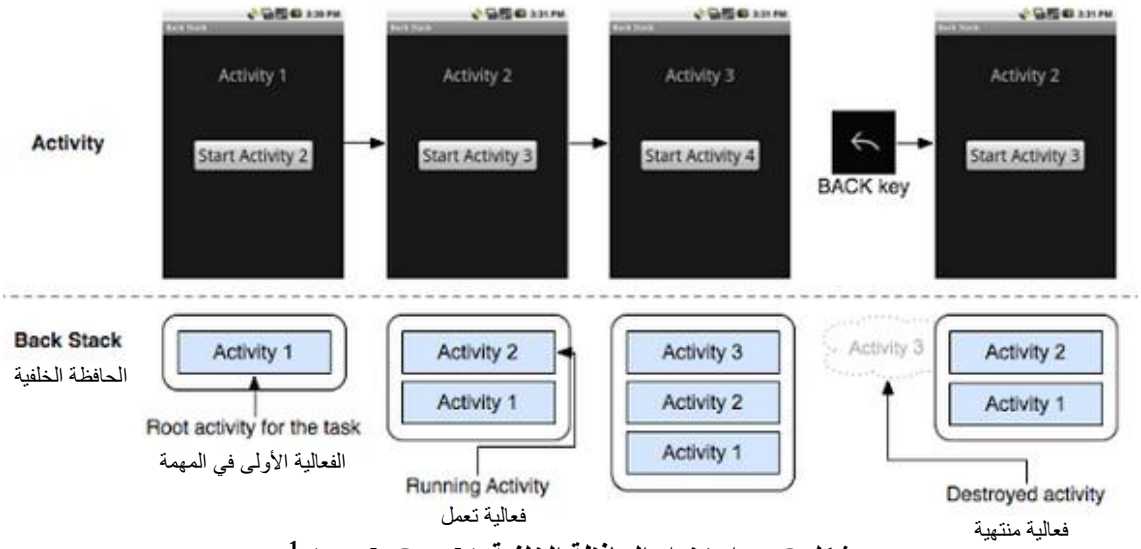


شكل 1-1: دورة حياة الفعالية (Activity Life Cycle) [1]

في كل مرة يتم فيها تشغيل فعالية جديدة، يتم ايقاف الفعالية السابقة. يقوم نظام الأندرويد بحفظ وترتيب الفعاليات في حافظة خلفية تسمى Back Stack كما هو موضح في شكل 2-1. كل فعالية جديدة يتم تشغيلها يتم إضافتها إلى مقدمة الحافظة تليها الفعاليات السابقة. الفعالية (Activity) التي في مقدمة الحافظة هي فقط تكون في وضع التشغيل running، بينما تكون كل الفعاليات التالية لها متوقفة عن العمل. عند إنهاء الفعالية التي في المقدمة عن طريق النقر على زر Back يتم إيقافها وإزالتها من الحافظة، وتصبح الفعالية التالية لها في مقدمة الحافظة الخلفية

[1] Activity, Android Developer, http://developer.android.com/reference/android/app/Activity.html

لتنتقل إلى وضع التشغيل. بالرجوع إلى مثال التطبيق الإخباري، عن تفعيل الفعالية الخاصة بأي خبر تنتقل هذه الفعالية إلى مقدمة الحافظة الخلفية، وتتوقف الفعالية الرئيسية عن العمل لتصبح في مؤخرة الحافظة الخلفية. عند انتهاء المستخدم من استخدام الفعالية الفرعية والنقر على زر Back، يتم ازالة الفعالية من الحافظة لتعود الفعالية الرئيسية للواجهة وتصبح في بداية الحافظة.



شكل 2-1: استخدام الحافظة الخلفية (Back Stack) [1]

تمرين عملي (1-1)

يهدف هذا التمرين إلى تتبع دورة حياة الفعالية (Activity Life Cycle) والتسلسل المستخدم في تنفيذ دوال الاتصال الراجع (Callback Methods). سنقوم في هذا التمرين بإنشاء فعالية بدون أي عناصر في واجهة، ومن ثم سنكتب كود طباعة في بعض دوال الاتصال الراجع. كود الفعالية موضح بالأسفل:

```
public class MainActivity extends Activity {

     @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          Toast.makeText(this, "onCreate",
Toast.LENGTH_SHORT).show();
     }
     @Override
     protected void onPause() {
```

[1] Android for 'Starters', http://techiezjunkyard-android.blogspot.com/2012/01/task-and-back-stack_8919.html

```
          super.onPause();
          Toast.makeText(this, "onPause",
Toast.LENGTH_SHORT).show();
     }
     @Override
     protected void onResume() {
          super.onResume();
          Toast.makeText(this, "onResume",
Toast.LENGTH_SHORT).show();
     }
     @Override
     protected void onStop() {
          super.onStop();
          Toast.makeText(this, "onStop",
Toast.LENGTH_SHORT).show();
     }
     @Override
     protected void onDestroy() {
          super.onDestroy();
          Toast.makeText(this, "onDestroy",
Toast.LENGTH_SHORT).show();
     }
}
```

قم بتشغيل التطبيق وتتبع ترتيب الرسائل التي يتم طباعتها.

بعد إكتمال تشغيل الفعالية وتوقف طباعة الرسائل، إنقر على زر Back لانهاء الفعالية، ولاحظ تسلسل الرسائل التي يتم طباعتها في هذه الحالة.

ملاحظة: لإنشاء رسالة ليتم طباعتها على الشاشة يمكن استخدام الدالة Toast.makeText(). هذه الدالة يمرر لها كائن من نوع Context وهو يمثل التطبيق الحالي ويمكن تمرير الفعالية الحالية. كما يمرر لها الرسالة المراد طباعتها ومدة العرض (Toast.LENGTH_SHORT أو Toast.LENGTH_LONG). بعد إنشاء الرسالة يتم طباعتها بتنفيذ الدالة Toast.show(). طباعة رسائل من نوع Toast سيتم استخدامه بكثرة في التمارين المذكورة في هذا الكتاب.

الوحدة الثانية:

بنية التطبيق (Application Structure)

يتعلم الطالب في هذه الوحدة:

- ✓ . المكونات الأساسية لتطبيق أندرويد والفرق بينها.
- ✓ . ملف الوثيقة (Manifest File) ومكوناته وتعديله لتهيئة التطبيق للعمل بشكل صحيح.
- ✓ . أنواع المصادر (Resources) الموجودة ضمن تطبيق أندرويد والوصول إليها.
- ✓ .التعامل بشكل صحيح مع تغير الإعدادات مثل تغير جهة العرض (رأسي - أفقي) وشاشة العرض.

لدراسة هذه الوحدة لابد من الإلمام بمفهوم الفعالية (Activity) ودورة حياتها (إرجع إلى الوحدة الأولى)، كذلك لابد من الإلمام بتصميم واجهات التطبيق (Layouts) باستخدام XML.

مكونات تطبيق أندرويد

يتكون تطبيق أندرويد من عدة مكونات تشمل بعض أو كل ما يلي:

- الفعاليات (Activities): تستخدم الفعاليات لعمل الواجهات التفاعلية. تم شرح الفعالية (Activity) في الوحدة السابقة، وسيتم استخدامها بكثرة في التطبيقات الموجودة في هذا الكتاب.
- الخدمات (Services): الخدمة هي مكون يعمل في الخلفية لتنفيذ عمليات يحتاج تشغيلها لفترة طويلة.الخدمة (Service) لاتوفر واجهة للمستخدم حيث تعمل في الخلفية بدون تدخل من المستخدم. على سبيل المثال، يمكن تفعيل خدمة تشغيل الملفات الصوتية في الخلفية بينما يقوم المستخدم بالتفاعل مع تطبيق مختلف، أوخدمة تزيل بيانات عبرالشبكة دون عرقلة تفاعل المستخدم مع التطبيقات الأخرى. لن يتم التطرق لبرمجة الخدمات في هذا الكتاب.
- مزودات المحتوى (Content Providers): مزودالمحتوى يتحكم في مشاركة قواعد بيانات أو الملفات. يمكنك تخزين البيانات في نظام الملفات أو في قاعدة بيانات SQLite، والوصول إليها من خلال مزود المحتوى (Content Provider). يمكن أيضاً لتطبيقات أخرى الاستعلام أوحتى تعديل البيانات (إذا كان مزود المحتوى يسمح بذلك). على سبيل المثال، يوفر نظام أندرويد مزود المحتوى الذي يدير معلومات الاتصال الخاصة بالمستخدم (Contacts Content Provider). على هذا النحو،أي التطبيق مع الأذونات المناسبة يمكنه الاستعلام عن جزء من المعلومات الخاصة بشخص معين. سيتم التطرق لمزودات المحتوى (Content Providers) في الوحدة السابعة من هذا الكتاب.
- مستقبلات النشر (Broadcast Receivers): مستقبل النشر هو المكون الذي يستجيب للرسائل المرسلة من النظام أو التطبيقات الأخرى. على سبيل المثال، عند إنخفاض مستوى شحن البطارية أو إعادة تشغيل الجهاز، يقوم النظام ببث رسائل للإبلاغ عن هذا الحدث. يمكن لمستقبل النشر (Broadcast Receiver) الاستجابة لهذه الرسائل وتنفيذ إجراءات ما. مستقبل النشر (Broadcast Receiver) لا يشتمل على واجهة مستخدم،

ولكنه قد يرسل تنبيها لمستخدم (Notification) عند حدوث حدث ما. سيتم التطرق لمستقبلات النشر (Broadcast Receiver) في الوحدة الثامنة من هذا الكتاب.

ملف الوثيقة (Manifest File)

كل تطبيق أندرويد يجب أن يحتوي على ملف الوثيقة باسم AndroidManifest.xml في المجلد الأساسي للتطبيق. يحتوي ملف الوثيقة على معلومات أساسية عن التطبيق حتى يتمكن النظام من تشغيله بشكل صحيح. الكود بالأسفل يوضح مثال لملف وثيقة (Manifest) لتطبيق ما. يحتاج المطور في كثير من الأحيان إلى إجراء تعديلات في هذا الملف ليعمل التطبيق بالشكل المناسب.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="ps.edu.ucas.FirstAndroidApp"
    android:versionCode="1"
    android:versionName="1.0" >
<!--الصلاحيات-->
<uses-permission
android:name="android.permission.WRITE_INTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="18" />

<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
<!-- الفعاليات -->
<activity android:name="
ps.edu.ucas.FirstAndroidApp.MainActivity"
android:label="@string/main_activity_title"
android:screenOrientation="portrait" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
```

```
</application>
</manifest>
```

سنقوم فيما يلي بشرح أهم المعلومات التي يحويها هذا الملف:

- وصف لمكونات التطبيق المختلفة من فعاليات (Activities) وخدمات (Services) وغيرها: كما ذكرنا في الوحدة السابقة، عند إنشاء أي فعالية يجب الإعلان عنها في ملف الوثيقة. أنظر المثال أعلاه حيث أن ملف الوثيقة يشتمل على فعالية باسم (MainActivity).
- الأذونات (Permission) والتي يحتاجها التطبيق من أجل الوصول إلى الأجزاء المحمية من النظام مثل الذاكرة الخارجية أو الاتصال بالانترنت.على سبيل المثال، هذه بعض الصلاحيات التي يتم إضافتها إلى ملف الوثيقة Manifest باستخدام الخاصية <uses-permission>:

```
<uses-permission
android:name="android.permission.WRITE_INTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.INTERNET" />
```

حيث تمنح هذه الأذونات التطبيق إمكانية الوصول للذاكرتين الداخلية والخارجية والإنترنت على التوالي.

- الحد الأدنى و الأعلى لمستوى الواجهة البرمجية (API Level) التي يتطلبها التطبيق للعمل بشكل صحيح. API Level هو رقم يحدد إصدار الواجهة البرمجية التي توفرها منصة أندرويد (Android Platform) ويتم تحديده في ملف الوثيقة (Manifest) من خلال الخاصية <uses-sdk>. من الهام تحديد قيم هذه الخاصية لأنها تحدد أرقام الإصدارات المتوافقة مع التطبيق. هناك قيمتان ضمن الخاصية <uses-sdk> يجب تحديدهما وهما: android:minSdkVersion و android:targetSdkVersion. القيمة الأولى تحدد أدنى واجهة برمجية يحتاجها التطبيق ليعمل بشكل صحيح. يقوم نظام أندرويد تلقائياً برفض تنصيب التطبيق إذا كانت الواجهة البرمجية المستخدمة من قبل النظام أقل من تلك المحددة في الخاصية android:minSdkVersion. عند تحديد هذه الخاصية يجب مراعاة طبيعة الأجهزة المتوافقة مع التطبيق وانتشارها في السوق: فمثلاً تحديد قيمة عالية للواجهة البرمجية يعني أن التطبيق الخاص بك لن يعمل على كل الأجهزة التي تعمل بالواجهات الأقدم والتي قد يستخدمها عدد كبير من الأجهزة. في المقابل فإن استخدام قيمة متدنية جدا يحرمك من استخدام الخصائص البرمجية الحديثة المضافة للواجهات البرمجية الأحدث.

قيمة android:targetSdkVersion تحدد الواجهة البرمجية التي يستهدفها التطبيق ، وهي تعني أن التطبيق تم اختباره على الواجهة البرمجية المحددة وأنه ليس هناك حاجة لأن يقوم النظام بأي عملية مواءة للتطبيق إذا توافقت الواجهة البرمجية للنظام مع واجهة التطبيق. لا يزال التطبيق قابل للعمل على الأنظمة القديمة (حتى الرقم المحدد في android:minSdkVersion) ولكن قد تقوم الانظمة القديمة بعمل مواءمة تلقائية للتطبيق مما قد يؤثر على بعض الإضافات الغيرمدعومة من قبل النظام.

الجدول 1-2 يوضح اسم منصة أندرويد (Code Name)، رقم الإصدار (Version) ورقم الواجهة البرمجية (API Level).

جدول 1-2: اسم منصة أندرويد (Code Name)، رقم الإصدار (Version) ورقم الواجهة البرمجية (API Level)[1]

API Level	Version	Code Name
API level 21	5.0	Lollipop
API level 19	4.4 - 4.4.4	KitKat
API level 18	4.3.x	Jelly Bean
API level 17	4.2.x	Jelly Bean
API level 16	4.1.x	Jelly Bean
API level 15, NDK 8	4.0.3 - 4.0.4	Ice Cream Sandwich
API level 14, NDK 7	4.0.1 - 4.0.2	Ice Cream Sandwich
API level 13	3.2.x	Honeycomb
API level 12, NDK 6	3.1	Honeycomb
API level 11	3.0	Honeycomb
API level 10	2.3.3 - 2.3.7	Gingerbread
API level 9, NDK 5	2.3 - 2.3.2	Gingerbread
API level 8, NDK 4	2.2.x	Froyo
API level 7, NDK 3	2.1	Éclair
API level 6	2.0.1	Éclair
API level 5	2.0	Éclair
API level 4, NDK 2	1.6	Donut
API level 3, NDK 1	1.5	Cupcake
API level 2	1.1	(no code name)
API level 1	1.0	(no code name)

- اتجاه الشاشة (screen orientation) للفعالية والتي تكون في أحد وضعين: الوضع الرأسي (portrait) والوضع الأفقي (landscape). إذا أردت تثبيت وضع العرض لفعالية ما (Activity) بحيث لا تستجيب لتغير وضع الجهاز إن كان رأسياً أو أفقياً، يمكن القيام ذلك باستخدام الخاصية android:screenOrientation من داخل الخاصية activity كما هو موضح في المثال التالي:

```
android:screenOrientation="portrait"
```

كما هو الحال في ملف XML الخاص بالواجهات، يمكن من ملف الوثيقة الوصول للنصوص المعرفة في ملف strings.xml. فمثلاً، في ملف الوثيقة (Manifest) الموضح مسبقاً: الخاصية label الموجودة ضمن العنصر

[1] Android Developers, http://developer.android.com/guide/topics/manifest/uses-sdk-element.html

application تستخدم قيمة المتغير app_name المعرف في ملف strings.xml. بالمثل، الخاصية label الموجودة ضمن الخاصية activity تستخدم قيمة المتغير main_activity_title المعرف أيضاً في ملف strings.xml.

إنشاء المصادر (Creating Resources)

عند بناء تطبيق أندرويد، يوصى دائماً بإبقاء المصادر التي يتعامل معها التطبيق(Application Resources) منفصلة عن الكود. من أنواع المصادر التي يتعامل معها التطبيق: العبارات النصية strings، الصور images، أنماط التصميم styles، ملفات هيكلة الواجهات layout وغيرها. هيكلية تطبيق الأندرويد مشابهة لما هو موضح في شكل 1-2، حيث أن المجلد src مخصص لحفظ ملفات الجافا، بينما مجلد res يستخدم لحفظ الأنواع المختلفة من المصادر، حيث يخصص لكل نوع من المصادر مجلد خاص داخل المجلد res. فمثلاً الصور تحفظ في مجلد drawable بينما تحفظ العبارات النصية strings داخل الملف strings.xml داخل المجلد values.

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            main.xml
            info.xml
        values/
            strings.xml
```

شكل 1-2: هيكلية تطبيق أندرويد

جدول 2-2 يوضح أهم مجلدات المصادر ونوعية البيانات التي تحفظ بها:

جدول 2-2: مجلدات المصادر (Resources) ضمن تطبيق أندرويد

مجلد المصدر	الاستخدام
drawable/	يحتوي على ملفات الصور
layout/	يحتوي على ملفات XML التي تحدد تصميم هيكلية الواجهات layout
menu/	يحتوي على ملفات XML التي تحدد تركيب القوائم المستخدمة في التطبيق
values/	يحتوي على ملفات XML تحدد قيم يتم استخدامها في التطبيق مثل الألوان والأرقام والعبارات النصية.
drawable/	يحتوي على ملفات الصور
layout/	يحتوي على ملفات XML التي تحدد تصميم هيكلية الواجهات layout

ملاحظة: المصادر التي يتم حفظها في المجلد res هي المصادر الإفتراضية التي يتم استخدامها في التطبيق. أحياناً قد تحتاج إلى مصادر مختلفة لتتلائم من الأوضاع المختلفة لعمل التطبيق. على سبيل المثال، التطبيق قد يعمل على أجهزة مختلفة الحجم، وبناءً على ذلك فإن تصميم الواجهات وأنواع الصور المستخدمة قد يختلف من جهاز لآخر. لتصميم التطبيق بحيث يتلاءم مع ظروف العرض المختلفة، لا بد من حفظ مصادر بديلة بالإضافة إلى تلك الإفتراضية ليتم استخدامها عند الحاجة لذلك. عند تشغيل التطبيق، يقوم نظام أندرويد تلقائياً باختيار المصادر المناسبة بما يتناسب مع إعدادات الجهاز.

لتحديد مصادر بديلة لإعدادات الجهاز المختلفة، يجب اضافة مجلد داخل المجلد res بالصيغة التالية:

`<resources_name>-<config_qualifier>`

حيث <resources_name> هو اسم مجلد المصدر مثل layout، values، drawable. <qualifier> يمثل الإعدادات التي يستخدم فيها المصدر. على سبيل المثال، في مجلد المصادر الموضح:

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

لاحظ أن المجلد drawable-hdpi يحوي الصور التي يتم استخدامها للشاشات ذات الدقة العالية وهو ما يعنيه المقطع hdpi.عند تشغيل التطبيق، يقوم النظام تلقائياً باختيار مجلد الصور المناسب لإعدادات الجهاز: حيث يتم استخدام الصور من المجلد drawable-hdpi إذا كان الجهاز ذو شاشة عالية الدقة، بينما يتم استخدام الصور من المجلد drawable لغير ذلك من الأجهزة.

مثال آخر على ذلك هو تحديد تصميم مختلف للواجهة إذا اختلف اتجاه الشاشة (Orientation) أو لغة الجهاز. في هذه الحالة يتم إنشاء مجلد باسم layout-port يحوي تصميم الواجهات للوضع الرأسي (portrait) أو مجلد باسم layout-land يحوي تصميم الواجهات للوضع الأفقي أو layout-ar ليحوي تصاميم واجهات متناسبة مع اللغة العربية.

الوصول للمصادر (Accessing Resources)

بعد إنشاء المصادر، قد تحتاج للوصول إليها أثناء بناء التطبيق. هناك طريقتين للوصول للمصادر: الوصول من خلال الكود أو من خلال XML.

- الوصول من خلال الكود:

كل المصادر التي يتم إنشاؤها يقوم النظام تلقائياً بإنشاء صيغ معرفة لها ID وحفظها في ملف خاص اسمه R. الصيغ المعرفة تأخذ الشكل التالي:

R.<resource_type>.<resource_name>

حيث <resource_type> يمثل نوع المصدر مثل string، layout، drawable وغيره، بينما <resource_name> تعني اسم المصدر. فمثلاً، المعرف التالي: R.string.activity_title يستخدم للوصول للعبارة النصية string التي تحمل اسم activity_title. المعرف R.drawable.background يستخدم للوصول لملف الصورة الذي يحمل اسم background داخل المجلد drawable. الأمثلة التالية توضح كيفية الوصول إلى المصادر برمجياً من خلال الكود:

مثال 1:الوصول إلى ملف صورة باسم myimage لعرضها في عنصر واجهة من نوع ImageView

```
imageView.setImageResource(R.drawable.myimage);
```

مثال 2: تغيير عنوان الفعالية (Activity) باستخدام العبارة النصية title

```
getWindow().setTitle(getResources().getText(R.string.title));
```

مثال 3: تحديد ملف هيكلية الواجهة باسم main_screen عند إنشاء الفعالية

```
setContentView(R.layout.main_screen);
```

- الوصول من خلال XML

عند إنشاء ملفات XML للواجهات وغيرها، يمكن اعطاء قيم لعناصر XML باستخدام قيم معرفة في ملفات أخرى. الصيغة العامة للوصول لأي مصدر من خلال XML تأخذ الشكل التالي:

@<resource_type>/<resource_name>

حيث <resource_type> يمثل نوع المصدر مثل string، layout، drawable وغيره، بينما <resource_name> تعني اسم المصدر.

المثال التالي يوضح كيف يتم إعطاء قيمة للنص المعروض على الزر Button باستخدام عبارة نصية string معرفة باسم submit (أنظر الخاصية android:text):

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/submit" />
```

المثال التالي يوضح إعطاء قيم للعنصر EditText باستخدام مصادر معرفة في أماكن أخرى (أنظر الخاصيتين: android:textColor و android:text):

```
<EditText
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="@string/title" />
```

مجلد الممتلكات (Assets Folder)

يوفر نظام أندرويد مجلد آخر ضمن مجلدات التطبيق حيث يمكن حفظ الملفات المختلفة. يسمى هذا المجلد assets. الفرق بين مجلد assets والمجلد res أن النظام لا يقوم بإنشاء أرقام معرفة ID للملف الموجودة في المجلد assets. يستخدم مجلد assets لحفظ الملفات التي لايمكن حفظها في ملف res مثل ملفات بالامتداد txt، ملفات الصوت (.wav, .mp3, .mid) وغيرها. يتم الوصول إلى الملفات في مجلد assets عن طريق الكائن AssetManager والذي يمكن الوصول إليه بتطبيق الدالة getAssets() داخل الفعالية (Activity). يوفر AssetManage عدة دوال للتعامل مع الملفات مثل list() لاستعراض الملفات و open() لفتح ملف ما. الكود التالي يوضح كيفية قراءة محتوى ملف text.txt موجود في مجلد assets وطباعته كرسالة.

```
public class AssetsActivity extends Activity {

     @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          try {
               InputStream is =
getAssets().open("text.txt");
               int size = is.available();
               byte[] buffer = new byte[size];
               is.read(buffer);
               is.close();
               String content = new String(buffer);
               Toast.makeText(getApplicationContext(),
content, Toast.LENGTH_LONG).show();
          } catch (IOException e) {
               e.printStackTrace();
          }
     }
}
```

التعامل مع تغيير الإعدادات (Handling Configuration Changes)

إعدادات الجهاز قد تتغير في أي وقت مثل تغير اتجاه العرض (أفقي – رأسي) أو تغير اللغة. عندما تحصل مثل هذه التغييرات، يقوم نظام أندرويد بإلغاء الفعالية الحالية (Activity) وإنشاء واحدة جديدة عن طريق تنفيذ الدالة onDestroy() ومن ثم onCreate(). إعادة تشغيل الفعالية يهدف إلى جعل التطبيق يتكيف مع التغيير الجديد باستخدام المصادر والواجهات التي تتلاءم مع الإعدادات الجديدة. عند إعادة تشغيل الفعالية يمكن استرجاع حالتها وبياناتها عن طريق تطبيق الدالة onSaveInstanceState() أو onPause() لحفظ حالة الفعالية قبل إيقافها، ومن ثم تطبيق الدالة onRestoreInstanceState() أو onStart() لاسترجاع البيانات.

يمكن تثبيت بعض الإعدادات للتطبيق بحيث لا يتم إعادة تشغيل الفعالية (Activity) تلقائياً عند تغيره. فمثلاً، يمكن إضافة الخاصية android:screenOrientation ضمن خصائص الفعالية (Activity) في ملف الوثيقة Manifest وذلك لتثبيت طريقة عرض التطبيق في وضع رأسي (أو أفقي) مهما تغير وضع الجهاز. كذلك من الممكن إيقاف إعادة تشغيل الفعالية عند حصول تغيير بالإعدادات بحيث يتم التعامل مع هذا التغيير برمجياً. يتم ذلك بإضافة الخاصية android:configChanges ضمن خصائص الفعالية (Activity) في ملف الوثيقة (Manifest) كما هو موضح:

```
<activity android:name=".MyActivity"
android:configChanges="orientation|keyboardHidden"
          android:label="@string/app_name">
```

في هذه المثال يتم التصريح بأن الفعالية ستتولى التعامل برمجياً مع تغيير اتجاه العرض orientation أو في حالة إخفاء لوحة المفاتيح، وذلك دون الحاجة لإعادة تشغيل الفعالية. في هذه الحالة يجب تطبيق الدالة onConfigurationChanges() داخل الفعالية والتي يتم تنفيذها تلقائياً عند حصول أي تغيير في الإعدادات. يمكن كتابة كود في هذه الدالة لتنفيذ الإجراء المطلوب كما بالمثال التالي:

```
@Override
public void onConfigurationChanged(Configuration
newConfig){
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
if(newConfig.orientation
==Configuration.ORIENTATION_LANDSCAPE){

Toast.makeText(this,"landscape",Toast.LENGTH_SHORT).show();
}elseif(newConfig.orientation
==Configuration.ORIENTATION_PORTRAIT){

Toast.makeText(this,"portrait",Toast.LENGTH_SHORT).show();
    }
}
```

في هذا المثال أعلاه يتم عرض رسالة تبين اتجاه عرض التطبيق وذلك عند حصول أي تغيير في الإعدادات. لاحظ أنه يتم تمرير الإعدادات التي تغيرت إلى الدالة.

في الغالب لن تحتاج لتطبيق هذه الدالة حيث يفضل إعادة تشغيل الفعالية (Activity) عن تغير الإعدادات وذلك حتى يتم استخدام المصادر المناسبة للاعدادات الجديدة تلقائياً. على سبيل المثال، عن وجود تصميم واجهات خاص في وضع العرض الرأسي portrait في مجلد layout-port، يقوم النظام تلقائياً باستخدام هذه الواجهات إذا تم تغيير اتجاه العرض للوضع الرأسي.

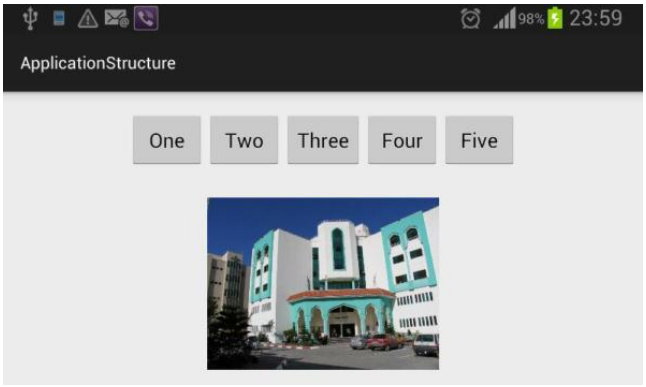
كذلك يمكن فحص إعدادات الجهاز برمجياً من داخل الفعالية (Activity) باستخدام الدالة getResources().getConfiguration(). المثال التالي يوضح كيفية معرفة اتجاه العرض برمجياً عند بدء تشغيل الفعالية (Activity):

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  if(this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE){
     Toast.makeText(this, "landscape",
Toast.LENGTH_SHORT).show();
  }else
if(this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_PORTRAIT){
     Toast.makeText(this, "portrait",
Toast.LENGTH_SHORT).show();
  }
…
}
```

تمرين عملي (1-2)

يهدف هذا التطبيق البسيط إلى التمرن على بعض المفاهيم الواردة في هذا الفصل مثل تغيير إعدادات التطبيق من خلال ملف الوثيقة (Manifest)، الوصول للمصادر وتحديد مصادر بديلة لتلائم الإعدادات المختلفة للتطبيق.

التطبيق له واجهتين كما هو موشح بشكل 2-2 : الواجهة الأفقى تعمل في وضع العرض الأفقي حيث تتكون من مجموعة من الأزرار المعروضة أفقياً وأسفلها صورة لمبنى الكلية الجامعية. الواجهة الثانية تعمل في وضع العرض الرأسي وفيها يتم عرض الأزرار رأسياً، ويتم عرض صورة لشعار الكلية الجامعية. يتم اختيار الواجهة وتغيير الصورة تلقائياً بناء على وضع العرض.



شكل 2-2: واجهتا التطبيق في الوضعين الرأسي (Portrait) والأفقي (Landscape)

الخطوات التالية توضح مراحل بناء التطبيق:

أولاً: إنشاء الواجهات: بناء على أن التطبيق له واجهتين مختلفيتين، نقوم بإنشاء ملفي XML لتصميم الواجهة: أحد الملفين مخصص للواجهة بالوضع الرأسي ونضعه في المجلد layout، والملف الآخر يحمل نفس الاسم ولكن في مجلد آخر بالاسم layout-land. لاحظ أن اسم المجلد يحدد الإعدادات التي يستخدم عندها ملف الواجهة، فالمجلد الذي ينتهي اسمه بـ land يستخدم في وضع العرض الأفقي (Landscape) بينما يستخدم المجلد الآخر في غير ذلك. ملفات التصميم موضحة بالأسفل:

أولاً: تصميم الواجهة الخاصة بالوضع الرأسي

```
// layout/activity_main.xml
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```
tools:context="com.example.applicationstructure.MainActivit
y" >
<Button
        android:id="@+id/buttonOne"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="@string/buttonOne" />
<Button
        android:id="@+id/buttonTwo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/buttonOne"
        android:text="@string/buttonTwo" />
<Button
        android:id="@+id/buttonThree"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/buttonTwo"
        android:text="@string/buttonThree" />
<Button
        android:id="@+id/buttonFour"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/buttonThree"
        android:text="@string/buttonFour" />
<Button
        android:id="@+id/buttonFive"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/buttonFour"
        android:text="@string/buttonFive" />
<ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@+id/buttonFive"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="24dp"/>
</RelativeLayout>
```

ثانياً: تصميم الواجهة الخاصة بالوضع الأفقي

```
// layout-land/activity_main.xml
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

tools:context="com.example.applicationstructure.MainActivit
y" >
<Button
        android:id="@+id/buttonOne"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/buttonTwo"
        android:text="@string/buttonOne" />
<Button
        android:id="@+id/buttonTwo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/buttonOne"
        android:layout_toLeftOf="@+id/buttonThree"
        android:text="@string/buttonTwo" />
<Button
        android:id="@+id/buttonThree"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/buttonOne"
        android:layout_centerHorizontal="true"
        android:text="@string/buttonThree" />
<Button
```

```
        android:id="@+id/buttonFour"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignBottom="@+id/buttonOne"
        android:layout_toRightOf="@+id/buttonThree"
        android:text="@string/buttonFour" />
<Button
        android:id="@+id/buttonFive"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/buttonOne"
        android:layout_toRightOf="@+id/buttonFour"
        android:text="@string/buttonFive" />
<ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/buttonTwo"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="24dp"/>
</RelativeLayout>
```

ثانياً: إنشاء الفعالية (Activity): الكود بالأسفل يوضح الفعالية (MainActivity) المستخدمة في التطبيق:

```
1:  public class MainActivity extends Activity {
2:
3:  @Override
4:  protected void onCreate(Bundle savedInstanceState) {
5:     super.onCreate(savedInstanceState);
6:     setContentView(R.layout.activity_main);
7:     ImageView image = (ImageView)
8:     this.findViewById(R.id.imageView);
9:    if(this.getResources().getConfiguration().orientation
10:    == Configuration.ORIENTATION_LANDSCAPE)
11:       image.setImageResource(R.drawable.ucas_building);
12:   else if(this.getResources().getConfiguration().
13:   orientation == Configuration.ORIENTATION_PORTRAIT)
14:       image.setImageResource(R.drawable.ucas_logo);
15:   }
16: }
```

لاحظ أنه في الدالة onCreate() (عند بداية تشغيل الفعالية) يتم إنشاء الواجهة عن طريق الدالة setContentView() والتي يمرر لها المعرف الخاص بملف الواجهة وهو R.layout.activity_main (أنظر سطر رقم 6). لاحظ أيضاً أن هذا المعرف يمكن استخدامه للوصول لكلا ملفي الواجهة. يقوم النظام تلقائياً باختيار الملف المناسب بناءً على وضع العرض: فإذا كان وضع العرض أفقياً (Landscape) يتم استخدام ملف الواجهة activity_main.xml الموجود في المجلد layout-land، بينما إذا كان وضع العرض رأسياً (Portrait) يتم استخدام ملف الواجهة activity_main.xml الموجود في المجلد layout.

بعد ذلك يتم فحص وضع العرض برمجياً عن طريق الدالة getResources().getConfiguration() وبناءً عليه يتم تغيير الصورة (أنظر الكود من سطر 9 إلى 15). لاحظ أيضاً أن الوصول للصورة تم من خلال المعرف:R.drawable.<image_name>.

الوحدة الثالثة:

التعامل مع عناصر الواجهة برمجياً

(Interacting with UI Components Programmatically)

يتعلم الطالب في هذه الوحدة:

- ✓ معالجة أحداث التفاعل مع واجهة المستخدم.
- ✓ الوصول لعناصر واجهة المستخدم برمجياً والتعامل مع أنواع عناصر الواجهة المختلفة.
- ✓ التعامل مع قائمة العرض (ListView) وتعبئتها باستخدام أنواع المحولات المختلفة (Adapter).

لدراسة هذه الوحدة لابد من الإلمام بمفهوم الفعالية (Activity) ودورة حياتها (إرجع إلى الوحدة الأولى)، التعامل مع المصادر (إرجع إلى الوحدة الثانية)، كذلك لابد من الإلمام بعناصر الواجهة الأساسية وكذلك تصميم واجهات التطبيق (Layouts) باستخدام XML.

درست في مساق سابق إنشاء عناصر الواجهة المختلفة مثل TextView، EditText، Button وغيرها من العناصر. كذلك تعلمت طرق الهيكلة المختلفة Layouts لتنظيم عرض عناصر الواجهة (لمراجعة هذه المفاهيم يمكن الرجوع إلى الفصل الثالث في المرجع رقم 1 أو الفصل الثالث في المرجع رقم 7). سنتعلم الآن طريقة الوصول لعناصر الواجهة برمجياً والتفاعل مع الأحداث UI Events. كذلك سنتطرق للتعامل مع بعض عناصر الواجعة شائعة الإستخدام مثل القائمة (ListView) وطريقة عرض البيانات المختلفة بها باستخدام المحولات Adapters.

معالجة أحداث الوجهة برمجياً (Handling UI Events)

يقصد بأحداث واجهة المستخدم UI Events هي الأحداث الناتجة عن تفاعل المستخدم مع مكونات الواجهة مثل نقر الأزرار أو لمس الشاشة.

للتعامل مع أي حدث يحصل على أي عنصر في الواجهة يجب أولا الوصول لهذا العنصر من داخل الفعالية، وذلك يتم باستخدام الدالة findViewById() والتي يمرر لها الرقم المعرف للعنصر (له صيغة: R.id.<element_name>. لكل عنصر في الواجهة رقم خاص يختلف عن أي معرف للعناصر الأخرى. يتم إنشاء هذا المعرف تلقائياً بعد إدارج العنصر في ملف التصميم layout.xml. تقوم دالة findViewById() بإرجاع مؤشر لكائن من نوع View وهو نوع فئة الأب (Superclass type) لكل عناصر الواجهة. نقوم بعمل casting للعنصر المرجع وذلك لتحويله للنوع المطلوب. فمثلاً، للوصول للزر button1 يمكن استخدام الأمر التالي من داخل الفعالية (Activity). بعد الحصول على مؤشر للكائن الخاص بالزر، يمكن التعامل معه باستخدام الدوال المختلفة التي توفرها الفئة Button Class.

```
Button button = (Button) this.findViewById(R.id.button1);
button.setText("Click here");
```

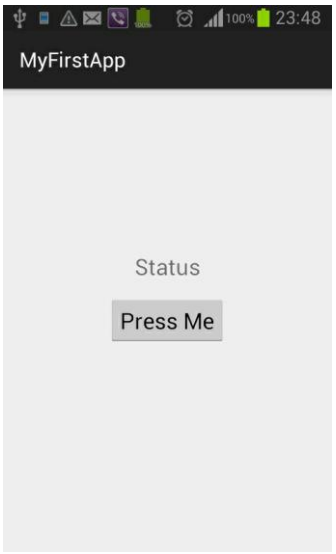
بعد الوصول لعناصر الواجهة برمجياً، يمكن التقاط الحدث (تفاعل المستخدم مع العنصر) أول وقوعه عن طريق تسجيل ما يسمى بالمستمع (Listener) والذي يعلمك تلقائياً بحصول حدث ما بتنفيذ دالة راجعة callback method. الفئة View، والتي تتفرع منها كل فئات عناصر واجهة المستخدم، تحتوي على مجموعة من الواجهات البرمجية (Interface) .كل واجهة برمجية (Interface) تحتوي على دوال مجردة والتي يمكن تطبيقها حسب الحاجة للاستماع للأحداث المختلفة.

الواجهة البرمجية (Interface) تحتوي فقط على دوال مجردة دون أي تطبيق، وعلى المستخدم تطبيق هذه الدوال بكتابة الكود المناسب. للتفاعل مع حدث معين على عنصر واجهة، يجب تسجيل واجهة مستمع جديدة لدى عنصر الواجهة وتطبيق الدوال الموجودة به لتنفيذ الإجراء المطلوب. فمثلاً، للاستجابة لحدث على زر معين Button، يجب على تسجيل واجهة مستخدم من نوع View.OnClickListener لدى الكائن Button (عن طريق تنفيذ الدالة setOnClickListener () ضمن الفئة Button، ومن ثم كتابة الكود الخاص بالدالة المجردة onClick() لتحديد الإجراء المطلوب تنفيذه عند الحدث). عند النقر على الزر يقول النظام تلقائياً بتنفيذ الدالة onClick() لينفذ الإجراء المطلوب.

```
button.setOnClickListener(
     new Button.OnClickListener() {
          public void onClick(View v) {
          // Action to be performed when button is clicked
          }
     }
);
```

تمرين عملي (1-3)

لتوضيح كيفية التعامل مع الأحداث بمثال عملي، سنقوم بعرض الخطوات الكاملة لتطبيق بسيط واجهته موضحة بالشكل. تتكون الواجهة من زر مكتوب عليه "Press Me" وعنصر عرض من نوع TextView. الحدث الذي نود تنفيذه هو تغيير محتوى النص المعروض في TextView عند النقر على الزر PressMe.



شكل1-3 : واجهة التطبيق الخاص بالتمرين 1-3

ملف هيكلية الواجهة لهذا المثال موضح بالأسفل، حيث أنه يحدد خصائص العنصر Button والمعرف بالإسم myButton، والعنصر TextView والمعرف بالاسم myTextView.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
        android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/mybutton_string" />
<TextView
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/myButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="41dp"
        android:text="@string/mytextview_string"
```

```
android:textAppearance="?android:attr/textAppearanceLarge"
/>
</RelativeLayout>
```

بعد تصميم الواجهة، يجب التعديل على كود الفعالية (Activity) وذلك لتسجيل مستمع للحدث (Listener) ليقوم بتنفيذ الإجراء المطلوب عن النقر على الزر. كود الفعالية المعدل موضح بالأسفل:

```
1:  public class MainActivity extends Activity {
2:  @Override
3:  protected void onCreate(Bundle savedInstanceState) {
4:     super.onCreate(savedInstanceState);
5:     setContentView(R.layout.activity_main);
6:  }
7:  @Override
8:  protected void onStart(){
9:     super.onStart();
10:  Button button = (Button)findViewById(R.id.myButton);
11:     button.setOnClickListener(
12:       new Button.OnClickListener() {
13:            public void onClick(View v) {
14:              TextView myTextView = (TextView)
15:              findViewById(R.id.myTextView);
16:              myTextView.setText("Button clicked");
17:            }
18:       }
19:     );
20:  }
21: }
```

لاحظ أن الكود المسؤول عن التعامل مع الحدث تمت كتابته في الدالة ()onStart (أنظر سطر رقم 9). تم اختيار الدالة ()onStart لأنه عند تنفيذها يكون قد اكتمل إنشاء عناصر الواجهة ومن ثم أصبح في الإمكان الوصول للعناصر الموجودة بها عن طريق الدالة ()View.findViewById(أنظر سطر رقم 15). إنشاء واجهة المستخدم يتم في الدالة ()onCreate، وتصبح الفعالية (Activity) مرئية عند تنفيذ الدالة ()onStart (راجع دورة حياة الفعالية وتسلسل تنفيذ دوالة الاتصال الراجع callback). يمكن كتابة الكود أيضاً في الدالة ()onCreate ولكن بعد تنفيذ الدالة ()setContentView والمسؤولة عن إنشاء الواجهة.

بعد الوصول إلى الكائن myButton باستخدام الدالة ()findViewById، نضيف المستمع (Listener) للكائن myButton بتنفيذ الدالة () setOnClickListener، ونضيف الكود الخاص بالإجراء المطلوب داخل الدالة

onClick(). الإجراء المطلوب تنفيذه عند النقر على الزر هو تعديل النص المعروض في عنصر الواجهة myTextView ليصبح "Button Clicked". لتنفيذ هذا الإجراء يتم أولاً الوصول العنصر myTextView باستخدام الدالة findViewById()، ثم نعدل النص المعروض بتنفيذ الدالة setText().

التعامل مع الحدث عن طريقة ملف Layout

يمكن تنفيذ حدث معين عند النقر على الزر بطريقة أخرى وذلك بتعديل ملف هيكلية الواجهة layout بإضافة الخاصية android:onClick وإعطائها قيمة تمثل اسم الدالة التي سيتم تنفيذها تلقائياً عن النقر على الزر كما هو موضح في المثال التالي:

```
<Button
        android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/mybutton_string"
     android:onClick="myButtonClicked" />
```

بعد تعديل ملف الواجهة، يجب التعديل في ملف الفعالية (Activity) بإضافة الدالة myButtonClicked كما هو موضح بالمثال التالي: (يجب أن يكون اسم الدالة مطابق للإسم المحدد في ملف Layout).

```
public void myButtonClicked(View view){
TextView myTextView = (TextView)
findViewById(R.id.myTextView);
myTextView.setText("Button clicked");
}
```

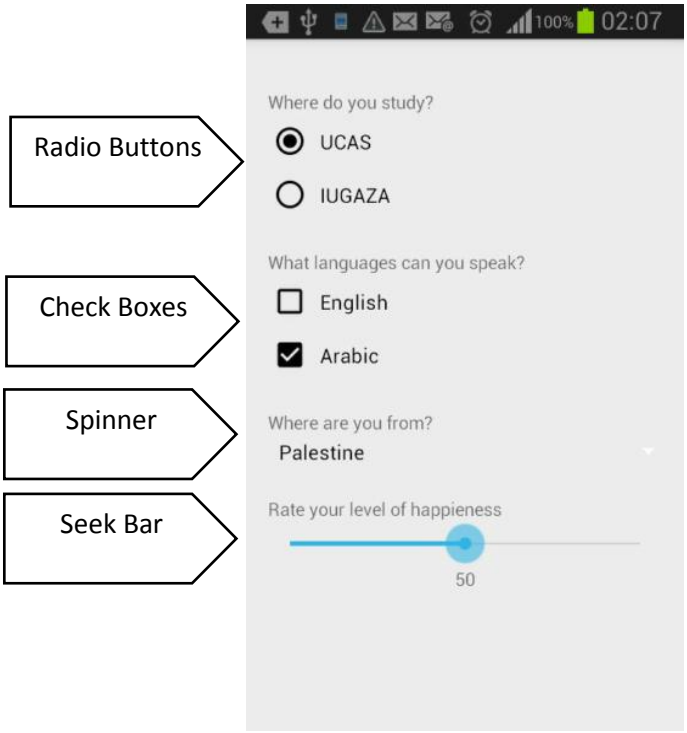
ملاحظة: يفضل استخدام الطريقة الأولى للاستماع لأحداث عناصر الواجهة وهي بإضافة المستمع (Listener) برمجياً وذلك لأنها تسمح بلاستماع لأحداث متنوعة عن طريقة دوال مختلفة مثل setOnClickListener()، setOnLongClickListener()، setOnTouchListener() وغيرها. كما أن هذه الطريقة تضمن فصل كامل ما بين مرحلة تصميم الواجهة، وهو ما يتم من خلال ملف هيكلية الواجهة، وبين مرحلة التعامل مع الأحداث والذي يتم برمجياً من خلال الفعالية (Activity).

في النهاية، نستعرض بعد مستمعات الأحداث (Event Listeners) التي توفرها عناصر الواجهة والتي يمكن استخدامها للاستماع لأنواع مختلفة من الأحداث، ومن ضمنها:

- onClickListener: ويستخدم لتنفيذ إجراء عند النقر ثم ترك عنصر الواجهة View. يتم كتابة كود الإجراء في الدالة ()onClick.
- onLongClickListener: ويستخدم لتنفيذ إجراء عند النقر لفترة على عنصر الواجهة. يتم كتابة كود الإجراء في الدالة ()onLongClick.
- onKeyListener: ويستخدم لتنفيذ إجراء عند الضغط على أحد المفاتيح في الجهاز بينما عنصر الواجهة View مفعّل من قبل المستخدم (has focus). يتم كتابة كود الإجراء في الدالة ()onKey.

تمرين عملي (2-3)

يتطرق هذا التمرين إلى عناصر واجهة متنوعة مثل زر الإنتقاء (RadioButton)، خانة الإختيار (CheckBox)، القائمة المنسدلة (Spinner) والشريط المنزلق (SeekBar) وكيفية معالجة الأحداث لهذه العناصر برمجياً وأنواع المستمعات (Listeners) المختلفة لهذا الغرض. الواجهة الخاصة بهذا التطبيق وأنواع العناصر المضافة موضحة في شكل 2-3.



شكل 2-3 : واجهة التطبيق الخاص بالتمرين 2-3

لتحديد مصفوفة النصوص (Stirng Array) التي يجب تعبئتها في القائمة المنزلقة (Spinner)، أضف المصفوفة countries_array إلى ملف strings كما هو موضح بالأسفل:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">UI Events</string>
<string name="action_settings">Settings</string>
<string-array name="countries_array">
<item>Palestine</item>
<item>Egypt</item>
<item>Algeria</item>
<item>Syria</item>
<item>Jordan</item>
<item>Saudi Arabia</item>
<item>Iraq</item>
</string-array>
</resources>
```

ملف تصميم الواجهة Layout الخاصة بهذا التمرين موضح بالأسفل:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.uievents.MainActivity" >
<TextView
    android:id="@+id/studyTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     android:layout_marginTop="20dp"
    android:text="Where do you study?" />
<RadioGroup
```

```
    android:id="@+id/radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<RadioButton
        android:id="@+id/ucasRadio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="UCAS" />
<RadioButton
        android:id="@+id/iugRadio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="IUGAZA" />
</RadioGroup>
<TextView
    android:id="@+id/languageTextView"
    android:layout_marginTop="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="What languages can you speak?" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
<CheckBox
        android:id="@+id/englishCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="English" />
<CheckBox
    android:id="@+id/arabicCheckBox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Arabic" />
</LinearLayout>
<TextView
    android:id="@+id/CountryTextView"
    android:layout_marginTop="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="Where are you from?" />
<Spinner
    android:id="@+id/countiresSpinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/countries_array"
     />
<TextView
    android:id="@+id/happienessTextView"
    android:layout_marginTop="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Rate your level of happieness" />
<SeekBar
    android:id="@+id/happienessSeekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/HappienessValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="" />
</LinearLayout>
```

لاحظ أن أزراء الانتقاء Radio Buttons تم جمعها في مجموعة واحدة RadioGroup وذلك حتى يتم اختيار عنصر واحد منها فقط. لاحظ أيضاً القائمة spinner وكيف تم تعبئتها بالمصفوفة المحددة مسبقاً في المصادر (ملف strings.xml) وذلك باستخدام الخاصية "android:entries="@array/countries_array (يمكن تعبئة القائمة spinner برمجياً باستخدام المحول Adapter والذي سيتم الحديث عنه لاحقاً في هذا الفصل). في أسفل الواجهة تم استخدام عنصر عرض TextView وذلك لعرض القيمة الراجعة من العنصر SeekBar. بعد تجهيز الواجهة، سيتم الآن معالجة أحداثها برمجياً داخل الفعالية (MainActivity) كما هو موضح:

```
1:  public class MainActivity extends Activity {
2:
3:  @Override
4:  protected void onCreate(Bundle savedInstanceState) {
5:     super.onCreate(savedInstanceState);
6:     setContentView(R.layout.activity_main);
7:  }
8:
9:  @Override
```

```
10: protected void onStart() {
11:    super.onStart();
12:    RadioGroup radioGroup = (RadioGroup)
13:    this.findViewById(R.id.radioGroup);
14:    radioGroup.setOnCheckedChangeListener(new
15:    android.widget.RadioGroup.OnCheckedChangeListener(){
16:
17:    @Override
18:    public void onCheckedChanged(RadioGroup group, int
19:    checkedId) {
20:       if(checkedId == R.id.ucasRadio)
21:          Toast.makeText(getApplicationContext(),
22:          "Choice : UCAS", Toast.LENGTH_SHORT).show();
23:       else if(checkedId == R.id.iugRadio)
24:          Toast.makeText(getApplicationContext(),
25:          "Choice : IUG", Toast.LENGTH_SHORT).show();
26:    }
27:  });
28:  CheckBox englishCheckBox = (CheckBox)
29:  this.findViewById(R.id.englishCheckBox);
30:  englishCheckBox.setOnCheckedChangeListener(new
31:  OnCheckedChangeListener(){
32:     @Override
33:     public void onCheckedChanged(CompoundButton
34:     buttonView, boolean isChecked) {
35:        Toast.makeText(getApplicationContext(),
36:        buttonView.getText()+" is "+ isChecked,
37:        Toast.LENGTH_SHORT).show();
38:    }
39:  });
40:  CheckBox arabicCheckBox = (CheckBox)
41:  this.findViewById(R.id.arabic(CheckBox));
42:  arabicCheckBox.setOnCheckedChangeListener(new
43:  OnCheckedChangeListener(){
44:    @Override
45:      public void onCheckedChanged(CompoundButton
46:      buttonView, boolean isChecked) {
47:
48:         Toast.makeText(getApplicationContext(),
49:         buttonView.getText()+" is "+ isChecked,
```

```
50:          Toast.LENGTH_SHORT).show();
51:       }
52:   });
53:
54:   Spinner spinner = (Spinner)
55:   this.findViewById(R.id.countiresSpinner);
56:   spinner.setOnItemSelectedListener(new
57:   OnItemSelectedListener(){
58:
59:     @Override
60:     public void onItemSelected(AdapterView<?> parent,
61:     View view, int position, long id) {
62:        Toast.makeText(getApplicationContext(),
63:        parent.getItemAtPosition(position).toString(),
64:        Toast.LENGTH_SHORT).show();
65:     }
66:     @Override
67:     public void onNothingSelected(AdapterView<?>
68:     parent){}
69:   });
70:   final TextView happienessValue = (TextView)
71:   this.findViewById(R.id.HappienessValue);
72:   SeekBar seekBar = (SeekBar)
73:   this.findViewById(R.id.happienessSeekBar);
74:   seekBar.setOnSeekBarChangeListener(new
75:   OnSeekBarChangeListener(){
76:     @Override
77:     public void onProgressChanged(SeekBar seekBar, int
78:     progress, boolean fromUser) {
79:       happienessValue.setText(String.valueOf(progress));
80:     }
81:
82:     @Override
83:     public void onStartTrackingTouch(SeekBar seekBar){}
84:
85:     @Override
86:     public void onStopTrackingTouch(SeekBar seekBar){}
87:   });
88: }
89:}
```

يتم معالجة الأحداث لكل عنصر كما يلي:

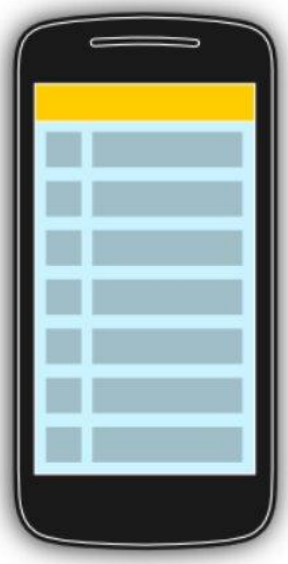
- Radio Buttons: لتحديد أي زر إنتقاء (Radio Button) تم اختياره، نقوم باستخدام المستمع RadioGroup.OnCheckedChangeListener وإضافته إلى مجموعة الأزرار RadioGroup (أنظر سطر رقم 14).عند حصول الحدث يتم تنفيذ الدالة ()onCheckedChanged والتي يمرر إليها رقم المعرف ID الخاص بالزر الذي تم اختياره حيث يتم طباعة رسالة تبين الإختيار (أنظر الكود من سطر 18 إلى 26).
- Check Boxes: للاستماع لحدث اختيار مربع الإختيار (CheckBox) نستخدم المستمع CompoundButton.OnCheckedChangeListener (أنظر سطور رقم 30 و 42)، وعند حصول الحدث يتم تنفيذ الدالة ()onCheckedChanged والتي يمرر إليها الكائن الخاص بمربع الإختياربالإضافة إلى قيمته (true-false) حيث يتم طباعة اسم مربع الاختيار وقيمته (أنظر الكود من سطر 33 إلى 38، ومن سطر 45 إلى 51).
- Spinner: نستخدم مستمع من نوع AdapterView.OnItemSelectedListener لمعالجة الحدث الخاص بهذا العنصر (أنظر سطر رقم 56). عند حصول الحدث يتم تنفيذ الدالة ()onItemSelected والتي من ضمن ما يمرر لها موقع العنصر الذي تم اختياره من القائمة. بمعرفة موقع العنصر يمكن الوصول له من القائمة عن طريق الدالة ()getItemAtPosition (أنظر الكود من سطر 60 إلى 65).
- Seek Bar: يتم معالجة الغيرات على الشريط المنزلق (SeekBar) عن طريق إضافة مستمع من نوع OnSeekBarChangeListener (أنظر سطر رقم 74). عند حصول الحدث يتم تنفيذ الدالة ()onProgressChanged والتي يمرر لها القيمة الحالية للشريط (0-100 مثلاً) بالإضافة إلى القيمة المنطقية fromUser والتي تحدد ما إذا كان تغير الشريط تم من المستخدم (fromUser=true) أو برمجياً (fromUser=false). في المثال الموضح يتم طباعة قيمة الشريط الممررة في عنصر من نوع TextView (أنظر الكود من سطر 77 إلى 80).

قائمة العرض (ListView)

عرض العناصر في قائمة هو أحد التصاميم الشائعة بكثرة في تطبيقات الهواتف النقالة. يشاهد المستخدم عدد من العناصر ويستطيع الانتقال لأعلى القائمة وأسفلها scroll up/down كما هو موضح بشكل 3-3 . عند إختيار أحد عناصر القائمة يتم عادة تنفيذ إجراء مثل فتح فعالية جديدة.

إضافة بيانات للقائمة (ListView)

لإضافة بيانات للقائمة (ListView) يتم عادةً استخدام المحول (Adapter)، وهو كائن وسيط بين عنصر العرض وهو القائمة (ListView) (أو أي عنصر واجهة يتفرع من الفئة AdaperView class) وبين البيانات المعروضة. يتحكم المحول (Adapter) في الوصول لبيانات القائمة، كما أنه مسؤول عن تحويل البيانات المدخلة للقائمة إلى عنصر عرض (View) يمكن تضمينه داخل القائمة. فمثلاً، عند إدخال مصفوفة من النصوص (Array String) لعرضها في القائمة، يقوم المحول (Adapter) بتحويل كل عبارة نصية في المصفوفة إلى كائن من نوع (TextView) والذي يتم إدراجه في القائمة، أي أن مصفوفة النصوص تتحول باستخدام المحول (Adapter) إلى مجموعة من عناصر العرض (TextView) المجمعة في قائمة من نوع (ListView).



شكل 3-3 : قائمة العرض (ListView) [1]

لاحظ أن محتوى القائمة قد يكون أكثر تعقيداً من مجرد عرض عبارات نصية. على سبيل المثال، قد يشتمل السطر الواحد في القائمة على صورة ونص وزر. في هذه الحالة يجب تصميم هيكلية الواجهة الخاصة بالسطر الواحد كملف Layout، ومن ثم نقوم بكتابة محول خاص (Custom Adapter) والذي يقوم باستقبال البيانات، الصور والنصوص، وإنشاء عنصر العرض (View) لعرضها بناءً على ملف layout. سنقوم لاحقاً في هذه الوحدة بتناول موضوع المحولات الخاصة (Custom Adapter).

المحولات الافتراضية (Default Adapter)

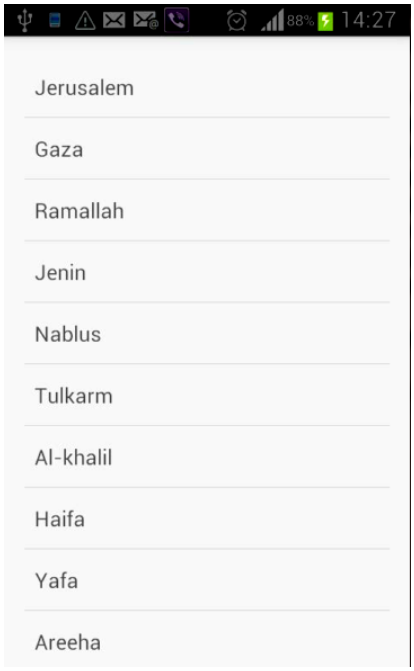
يوفر نظام أندرويد بعض المحولات الافتراضية، من أهمها ArrayAdapter و CursorAdapter. يستخدم ArrayAdapter لإدراجات البيانات الموجودة في مصفوفة Array أو List.المحول المخصص (CursorAdapter) يعالج البيانات المدخلة من قاعدة بيانات أو مزود المحتوى (Content Provider). كل هذه المحولات Adapters هي فئات فرعية (subclasses) من الفئة الأساسية BaseAdapter.

تمرين عملي (3-3)

يوضح هذا التمرين استخدام ArrayAdapter لتعبئة قائمة العرض (ListView) بمصفوفة من العبارات النصية. ArrayAdapter يتعامل مع مصفوفة من الكائنات objects حيث أن كل كائن سيمثل كسطر في القائمة (ListView).

[1] Using Lists in Android, atutorial by Lars Vogel, http://www.vogella.com/tutorials/AndroidListView/article.html

في هذا التمرين سنقوم بتعبئة قائمة (ListView) باسماء مدن فلسطينية كما هو موضح في شكل، وعند النقر على أي اسم يتم معالجة الحدث بطباعة رسالة على الشاشة.



شكل 4-3 : واجهة التطبيق الخاصة بالتمرين 3-3

ملف هيكلية الواجهة (Layout) الخاص بالبرنامج موضح بالأسفل، حيث تحتوي الواجهة على عنصر واحد فقط وهو القائمة (ListView):

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.simplelist.MainActivity" >
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
```

الكود الخاص بالفعالية MainActivity موضح في الأسفل:

```
1:  public class MainActivity extends Activity {
2:
3:   @Override
4:   protected void onCreate(Bundle savedInstanceState) {
5:        super.onCreate(savedInstanceState);
6:        setContentView(R.layout.activity_main);
7:        ListView listView = (ListView)
8:  this.findViewById(R.id.listView);
9:        String[] values = {"Jerusalem", "Gaza",
10: "Ramallah", "Jenin", "Nablus", "Tulkarm","Al-
11: khalil","Haifa","Yafa","Areeha"};
12:       ArrayList<String> listValues = new
13: ArrayList<String>();
14:       for(int i=0; i<values.length; i++)
15:            listValues.add(values[i]);
16:       ArrayAdapter<String> adapter = new
17: ArrayAdapter<String>(this,
18: android.R.layout.simple_list_item_1, listValues);
19:       listView.setAdapter(adapter);
20:       listView.setOnItemClickListener(new
21: OnItemClickListener(){
22:
23:           @Override
24:           public void onItemClick(AdapterView<?>
25: parent, View view, int position, long id) {
26:                Toast.makeText(getApplicationContext(),
27: parent.getItemAtPosition(position).toString(),
28: Toast.LENGTH_SHORT).show();
29:           }
30:       });
31:  }
32: }
```

لتعبئة قائمة العرض (ListView) يتم تنفيذ الخطوات التالية: يتم أولاً الوصول للقائمة عن طريقة الدالة findViewById() (أنظر سطر 7 و 8)، ثم نقول بإنشاء مصفوفة (List) تحتوي على أسماء المدن المراد إضافتها للقائمة (أنظر السطور من 9 إلى 15). لإدراج محتويات المصفوفة (List) في قائمة العرض (ListView) نستخدم ArrayAdapter ويتم إنشاءه كالتالي:

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, listValues);
```

حيث يمرر له ثلاث قيم هي: context (الفعالية الحالية)، المعرف الخاص بتصميم هيكلية الواجهة (android.R.layout.simple_list_item_1)، وقائمة البيانات (listValues). تحتاج قائمة العرض (ListView) إلى تصميم يحدد كيفية عرض كل سطر وخصائص العرض (التنسيق والهوامش والخطوط وغيره من الخصائص). يمكن تصميم شكل السطر في القائمة في ملف Layout مستقل، ومن ثم تمريره للمحول (Adapter) عند إنشائه. في هذا التمرين، تم تمرير أحد التصميمات الافتراضية التي توفرها بيئة أندرويد والتي تحمل الصيغة :android.R.layout.<layout_name> . هناك المزيد من التصميمات والتي يمكن تجربتها وملاحظة التغير في شكل القائمة (ListView).

بعد إنشاء المحول (Adapter) وإدخال البيانات إليه، نقوم بتمريره إلى القائمة (ListView) عن طريق الدالة ()setAdapter من خلال (ListView) (أنظر سطر رقم 19). كما ذكرنا مسبقاً فإنه المحول (Adapter) يقوم بتحويل البيانات المدخلة إلى عناصر Views ومن ثم تجميعها وعرضها في القائمة (ListView).

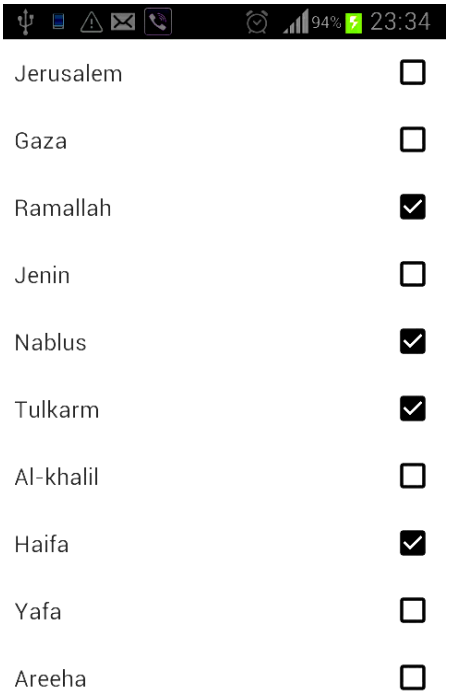
لمعالجة حدث النقر على أي عنصر من القائمة، قمنا بإضافة مستمع (Listener) من نوع OnItemClickListener (أنظر سطر رقم 20). عند حصول الحدث يتم تنفيذ الدالة ()onItemClicked والتي يمرر إليها موقع العنصر الذي تم اختياره من القائمة (ListView). يتم بعد ذلك الوصول و طباعة اسم العنصر الذي تم اختياره عن طريق الدالة ()getItemAtPosition (أنظر الكود من سطر 24 إلى 31).

لاحظ أن البيانات التي يتم تمريرها إلى المحول (Adapter) قد لا تكون بيانات نصية String، بل قد تكون كائن (object) من أي نوع. في حالة تمرير كائن غير نصي، يقوم المحول (Adapter) ضمنياً بتنفيذ الدالة ()toString لتمثيل الكائن كنص. لذلك احرص على تطبيق الدالة ()onString لأي كائن تقوم بإنشائه حتى تعمل القائمة (ListView) بعرض البيانات بالشكل الصحيح.

القائمة (ListView) في المثال السابق تسمح باختيار عنصر واحد من القائمة. يمكن تفعيل الإختيار المتعدد من القائمة عن طريق الدالة ()setChoiceModel بتنفيذ الأمر التالي:

```
listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

كما يوفر نظام أندرويد تصماميم افتراضية للقوائم متعددة الإختيار مثل android.R.layout.simple_list_item_multiple_choice والتي يمكن تمريرها للمحول كما فعلنا في التمرين السابق، حيث تصبح واجهة التطبيق السابق عن تطبيق خاصية الاختيار المتعدد كما بالشكل

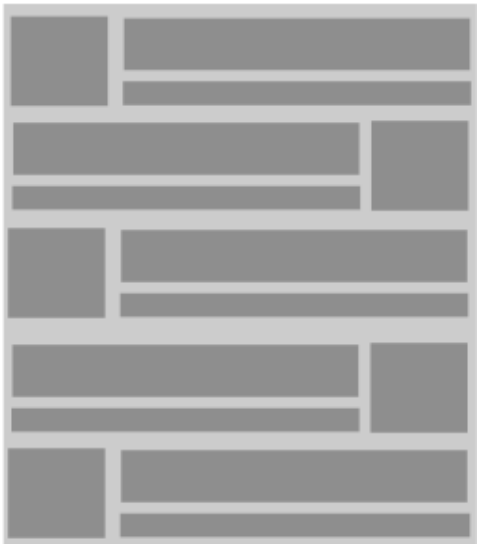


شكل 5-3 : الاختيار المتعدد من القائمة (ListView)

الجدير بالذكر أن نظام أندرويد يوفر فئة class باسم ListActivity، وهي فعالية ذو واجهة مستخدم إفتراضية تحتوي على قائمة (ListView). في حالة استخدام ListActivity بدلاً من Activity، لن تحتاج إلى تصميم ملف واجهة للتطبيق وإدراج قائمة جديدة (ListView) ، حيث يمكن الوصول إلى القائمة (ListView) الموجودة افتراضياً في الفعالية ListActivity عن طريقة الدالة () ListActivity.getListView.

المحول الخاص Custom Adapter

المحول الإفتراضي ArrayAdapter يوفر فقط إمكانية إضافة العناصر النصية من نوع String، وأي بيانات من نوع آخر يتم تمريرها للمحول ArrayAdapter يتم تحويلها تلقائياً لنص عن طريق الدالة ()toString. في كثير من الأحيان تحتاج لإضافة أنواع أخرى من البيانات للقائمة (ListView) مثل الصور. كذلك قد يكون السطر الواحد في القائمة مركباً حيث يتكون من أكثر من جزء كما هو موضح في الهيكلية الموضحة في شكل 6-3.



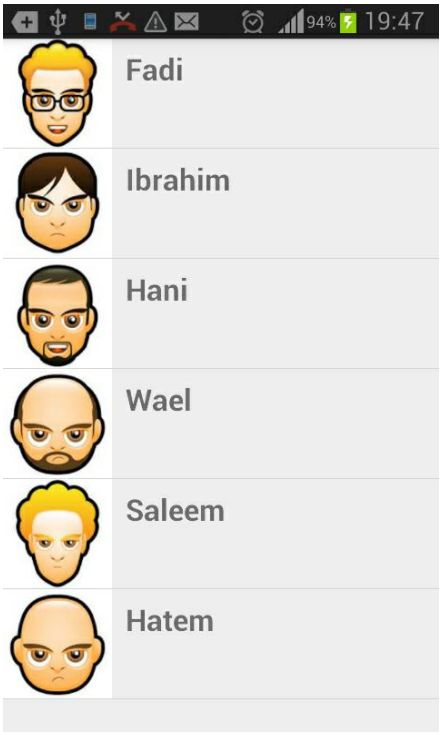
شكل 3-6 : قائمة عرض (ListView) حيث أن تصميم السطر يكون مركباً من مجموعة من العناصر[1]

في مثل هذه الحالات، نحتاج لإنشاء محول خاص (Custom Adapter) نحدد فيه تصميم الواجهة الخاصة بالسطر في قائمة العرض (ListView)، ونحدد كذلك كيف يتم معالجة البيانات المختلفة لعرضها في سطر القائمة. للقيام بذلك نقوم بإنشاء فئة (class) جديدة تمثل المحول الجديد بحيث تتفرع من الفئة BaseAdapter أو أي فئة متفرعة من BaseAdapter مثل ArrayAdapter، ثم نقوم بتطبيق الدالة ()getView والتي يتم من خلالها إنشاء الواجهة الخاصة بالسطر وعرض البيانات في عناصرها. التمرين التالي يوضح خطوات بناء واستخدام المحول الخاص (Custom Adapter) ضمن تطبيق بسيط.

تمرين عملي (3-4)

في هذا التمرين سيتم استخدام القائمة (ListView) بالإضافة لمحول خاص (Custom Adapter) لإنشاء واجهة التطبيق الموضحة بالشكل 3-7 : حيث تتكون الواجهة من قائمة (ListView) كل سطر فيها من مكون اسم شخص وصورته، وعند النقر على أي سطر يتم طباعة رسالة باسم الشخص.

[1] Using lists in Android, A tutorial by Lars Vogel, http://www.vogella.com/tutorials/AndroidListView/article.html



شكل 7-3: واجهة التطبيق الخاص بالتمرين 4-3

يتم في البداية تصميم واجهة السطر والتي تتكون من عنصر لعرض الصورة ImageView بالإضافة لعنصر نصي TextView لعرض الإسم. الملف التالي my_listview_layout.xml يوضح تصميم هيكلية الواجهة.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="80dp"
     android:layout_height="80dp"
        android:src="@drawable/ic_launcher" />
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="44dp"
```

```
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_marginLeft="10dp"
        android:gravity="center_vertical"/>
</LinearLayout>
```

بعد ذلك يتم إنشاء المحول الخاص (Custom Adapter) والذي سيستخدم في الواجهة في الشكل 7-3 تصميمها لبناء القائمة (ListView). الكود الموضح بالأسفل يوضح الفئة (class) الخاصة بالمحول الخاص (Custom Adapter):

```
1:  public class MyCustomAdapter extends
2:  ArrayAdapter<String>{
3:   Context context;
4:   List<String> names;
5:   List<Integer> photos;
6:
7:   public MyCustomAdapter(Context context, List<String>
8:   names, List<Integer> photos) {
9:        super(context, R.layout.my_listview_layout,
10:  names);
11:       this.names = names;
12:       this.photos = photos;
13:       this.context = context;
14:  }
15:
16:  public View getView(int position, View view, ViewGroup
17:  parent){
18:       if(view == null){
19:            LayoutInflater inflater = (LayoutInflater)
20: context.getSystemService(
21: Context.LAYOUT_INFLATER_SERVICE);
22:            view =
23: inflater.inflate(R.layout.my_listview_layout, parent,
24: false);
25:            TextView tv = (TextView)
26: view.findViewById(R.id.textView1);
27:            ImageView iv = (ImageView)
28: view.findViewById(R.id.imageView1);
29:            tv.setText(names.get(position));
30:            iv.setImageResource(photos.get(position));
```

```
31:
32:             ViewHolder vh = new ViewHolder();
33:             vh.text = tv;
34:             vh.image = iv;
35:             view.setTag(vh);
36:        }else{
37:             ViewHolder vh = (ViewHolder) view.getTag();
38:             vh.text.setText(names.get(position));
39:
40:   vh.image.setImageResource(photos.get(position));
41:        }
42:        return view;
43:   }
44:
45:   static class ViewHolder{
46:        public TextView text;
47:        public ImageView image;
48:   }
49:}
```

كما تلاحظ فإن الفئة الجديدة MyCustomAdapter متفرعة من الفئة ArrayAdapter وتقوم بتطبيق الدالة getView().

من خلال الباني (Constructor) (أنظر السطور من رقم 7 إلى 14) يتم تمرير الكائن context (الفعالية مثلاً)، والمصفوفة الخاصة بالأسماء المراد عرضها (names) ثم مصفوفة الصور (photos) ممثلة بأرقام المعرفات IDs الخاصة بالصور. سيقوم المحول Adapter باستخدام هذه البيانات في بناء عناصر القائمة (ListView). لاحظ أن المحول ArrayAdapter يتطلب تمرير تصميم واجهة القائمة المعرفة بـ R.layout.my_listview_layout بالإضافة إلى قائمة من الجمل النصية String إلى الفئة الأم (superclass)وهي ArrayAdaper، وهو ما يتم من خلال الأمر التالي في الباني Constructor:

```
  super(context, R.layout.my_listview_layout, names);
```

الدالة الأساسية في المحول (Adapter) هي getView() (أنظر سطر رقم 16) والتي تحدد طريقة تحويل البيانات لكل سطر إلى عنصر عرض View ليكون ضمن القائمة (ListView). لاحظ أنه يتم استخدام كائن من نوع LayoutInflater لتحويل الواجهة الممثلة بملف XML إلى كائن نوع View. يمرر إلى الدالة inflate() المعرف الخاص بالواجهة المنشئة مسبقاً (R.layout.my_listview_layout) لتقوم بإرجاع كائن من نوع View (أنظر الكود من سطر 18 إلى 24).

يشتمل الكائن View داخله على عناصر الواجهة وهي ImageView و TextView. يتم بعد ذلك الوصول لكل من هذه العناصر عن طريق تنفيذ الدالة findViewById() على نطاق الكائن View (أنظر الكود من سطر 25 إلى 28).

```
TextView tv = (TextView) view.findViewById(R.id.textView1);
ImageView iv = (ImageView)
view.findViewById(R.id.imageView1);
```

بعد الوصول إلى عناصر السطر في القائمة (ListView)، يتم إضافة البيانات إليه (أنظر سطر رقم 29 و30). الدالة getView() يتم تنفيذها لكل سطر في القائمة (ListView)، وعند تنفيذها لأي سطر يمرر إليها رقم هذا السطر في المتغيرposition. فمثلاً، لعرض السطر الأول في القائمة يتم تنفيذ الدالة getView() تلقائياً ويمرر إليها position =0. لإضافة بيانات لسطر ما، نضيف البيانات الموجود في المصفوفات الخاصة بالأسماء والصور في المكان position. على سبيل المثال السطر الثالث في القائمة (ListView) حيث قيمة position تساوي 2 يأخذ البيانات الموجودة في المصفوفات من المكان 2. هذا يضمن أن يكون ترتيب العناصر في القائمة (ListView) مطابق لترتيبها في المصفوفات الخاصة بالبيانات. يتم ذلك من خلال السطرين التاليين في الكود:

```
tv.setText(names.get(position));
iv.setImageResource(photos.get(position));
```

وأخيراً، لاحظ أن الكود المرفق للمحول الخاص Cutom Adapter يستخدم ما يسمى بـ ViewHolder Pattern وذلك لتجنب تنفيذ الدالة findViewById() والتي تستنفذ وقت قد يؤثر على سرعة عرض (ListView) وسلاسة تحريكها. تعتمد هذه الفكرة على تخزين مؤشرات لعناصر الواجهة View في كائن من نوع ViewHolder (أنظر الكود من سطر 45 إلى 48) وذلك في أول مرة يتم فيها إن شاء الواجهة باستخدام LayoutInflater. ثم يتم إلحاق الكائن من نوع ViewHolder بعنصر العرض View الخاص بالواجهة عن طريق الدالة setTag()، وهو ما يتم من خلال الكود:

```
ViewHolder vh = new ViewHolder();
vh.text = tv;
vh.image = iv;
view.setTag(vh);
```

عندما يتم تنفيذ الدالة getView() ثانياً (كما في حالة عمل scroll لعرض عنصر تم عرضه مسبقاً)، فإن الدالة getView() لن تقوم بإنشاء عنصر الواجهة ثانياً باستخدام Layout Inflater لأنه قد تم إنشاؤه مسبقاً، حيث أن قيمة الكائن View المدخل للدالة getView() لا يساوي null في حالة إعادة عرض العنصر. في هذه الحالة لن يتم إنشاء الواجهة ثانية، ويتم استخدام الكائن ViewHolder الملحق بعنصر الواجهة View للوصول للعناصر TextView و ImageView بدون الحاجة لاستخدام الدالة getViewById() (أنظر الكود من سطر 36 إلى 41)، وهو ما يتم من خلال الكود التالي:

```
ViewHolder vh = (ViewHolder) view.getTag();
vh.text.setText(names.get(position));
vh.image.setImageResource(photos.get(position));
```

بعد إنشاء الفئة (class) الخاصة المحول (MyCustomAdapter) ، نقوم الآن ببناء الفعالية (Activity) والتي سيتم من خلالها إنشاء كائن من المحول MyCustomAdapter وتمرير البيانات المطلوب تعبئتها في القائمة (ListView) وهي أسماء وصور الأشخاص. الكود التالي خاص بهذه الفعالية:

```
1: public class MainActivity extends ListActivity {
2:
3:   @Override
4:   protected void onCreate(Bundle savedInstanceState) {
5:        super.onCreate(savedInstanceState);
6:        List<String> names = new ArrayList<String>();
7:        names.add("Fadi");
8:        names.add("Ibrahim");
9:        names.add("Hani");
10:       names.add("Wael");
11:       names.add("Saleem");
12:       names.add("Hatem");
13:
14:       List<Integer> photos = new ArrayList<Integer>();
15:       photos.add(R.drawable.face1);
16:       photos.add(R.drawable.face2);
17:       photos.add(R.drawable.face3);
18:       photos.add(R.drawable.face4);
19:       photos.add(R.drawable.face5);
20:       photos.add(R.drawable.face6);
21:
22:       MyCustomAdapter adapter = new
23: MyCustomAdapter(this, names, photos);
24:
25:       this.setListAdapter(adapter);
26:       this.getListView().setOnItemClickListener(new
27: OnItemClickListener(){
28:
29:            @Override
30:            public void onItemClick(AdapterView<?>
```

```
31: adapterView, View view, int position, long id) {
32:       Toast.makeText(getApplicationContext(),
33: adapterView.getItemAtPosition(position).toString(),
34: Toast.LENGTH_SHORT).show();
35:             }
36:       });
37:   }
38:}
```

في الكود السابق، يتم في الدالة onStart() انشاء القوائم Lists الخاص بالأسماء والصور (أنظر السطور من 6 إلى 20) (لاحظ أن الصور يتم الوصول إليها باستخدام المعرفات بالصيغة: R.drawable.<image_file_name>). يجب أن تكون الصور محفوظة بنفس الأسماء في مجلد drawable. لاحظ أيضاً أن الفعالية متفرعة من (ListActivity) وهو ما يعني أننا لسنا بحاجة لتصميم واجهة خاصة بالفعالية، حيث أن الواجهة الافتراضية تشتمل على قائمة (ListView). يتم الوصول لهذه القائمة عن طريق الدالة ListActivity.getListView() من داخل الفعالية.

يتم إنشاء كائن من نوع المحول MyCustomAdapter وتمرير البيانات له، ومن ثم تمرير هذا الكائن إلى القائمة (ListView) عن طريق الدالة setListAdapter() (أنظر الكود من سطر 22 إلى 25).

وفي النهاية يتم الاستماع لحدث النقر على عناصر القائمة ListView عن طريق مستمع من نوع OnItemClickListener حيث يتم طباعة الإسم الذي تم النقر عليه (أنظر الكود من سطر 26 إلى 37).

الوحدة الرابعة:

الربط بين الفعاليات باستخدام الأهداف

(Linking Activities Using Intents)

يتعلم الطالب في هذه الوحدة:

- ✓ .مفهوم الهدف (Inent) وأنواعة واستخداماته.
- ✓ تشغيل فعالية (Activity) من فعالية أخرى.
- ✓ التواصل بين الفعاليات وتبادل البيانات بينها.
- ✓ .مفهوم مرشح الهدف (Intent Filter) واستخداماته.

تطبيق أندرويد قد يتكون من أكثر من فعالية (Activity)، بحيث يتم الإنتقال من فعالية إلى أخرى أثناء استخدام التطبيق. فمثلاً، قد تكون هناك فعالية (Activity) تعرض قائمة عرض (ListView)، وعند اختيار أي عنصر من القائمة يتم تشغيل واجهة جديدة (فعالية) لعرض بيانات متعلقة بالعنصر الذي تم اختياره. في مثل هذه الحالات، يجب تشغيل فعالية من فعالية أخرى، وكذلك قد يلزم نقل البيانات بين الفعاليات. الربط بين الفعاليات في نظام أندرويد يتم باستخدام ما يسمى بـIntent أو الهدف (ترجمة حرفية). يتناول هذا الفصل مفهوم الهدف (Intent) وأنواعة المختلفة وطريقة تشغيل الفعاليات باستخدام (Intent)، ويتم تدعيم كل هذه المفاهيم بتمارين عملية.

الأهداف (Intents)

الـ (Intent) أو الهدف هو كائن (Object) يستخدم للتواصل بين مكونات التطبيقات لطلب إجراء معين أو لتبادل المعلومات. يمكن تخيل الهدف بأنه رسالة يتم إرسالها إلى التطبيقات لطلب إجراء ما أو لتبادل المعلومات. هناك ثلاث استخدامات أساسية للهدف (Intent) وهي كالتالي:

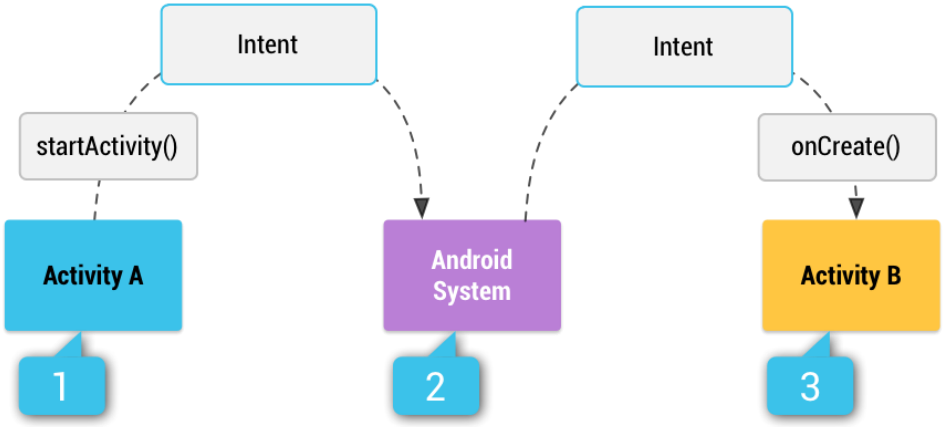
- تشغيل فعالية (Activity) واستقبال النتائج منها: الفعالية تمثل واجهة من واجهات التطبيق. يمكن تشغيل فعالية ما عن طريق تمرير هدف (Intent) إلى الدالة ()startActivity. في هذه الحالة يحدد الهدف (Intent) الفعالية المراد تشغيلها بالإضافة لأي بيانات أخرى تحتاجها الفعالية.
- تشغيل خدمة (Service): الخدمة (Service) هي مكون من مكونات تطبيق أندرويد يستخدم لتنفيذ مهام بدون واجهة للمستخدم، وغالباً ما تعمل الخدمة في الخلفية بدون تفاعل من المستخدم. يتم تشغيل خدمة معينة (Service) عن طريق الدالة ()startService والتي يمرر لها كائن من نوع الهدف (Intent) يحدد الخدمة المراد تشغيلها.
- إرسال منشور (Deliver Broadcast): المنشور (Broadcast) هي رسالة يتم ارسالها إلى تطبيقات مختلفة. على سبيل المثال، عن إعادة تشغيل الجهاز boot يقوم نظام أندرويد تلقائياً بارسال رسالة من نوع (Intent) وذلك لإعلام كل البرامج المعنية بحدث boot حتى تنفذ بناءً عليه إجراءات أخرى. أرسال المنشور Broadcast سيتم التطرق إليه في الوحدة الثامنة.

أنواع الهدف (Intent Types)

هناك نوعان للهدف (Intent)، وهما:

- الهدف الصريح (Explicit Intent) :وفيه يتم تحديد المكون المراد تشغيله (فعالية Activity أو خدمة Service أو منشور Broadcast). غالباً ما يتم استخدام الهدف الصريح لتشغيل مكون من مكون آخر في نفس التطبيق لأن الفئة (class) الخاصة بالمكون المراد تشغيله معروفة ويمكن الوصول إليها في نطاق التطبيق الواحد.
- الهدف المضمن (Implicit Intent): وفيه لا يتم تحديد المكون المراد تشغيله صراحةً، ويتم تحديد نوع الإجراء المراد تنفيذه بدلاً من ذلك. تحديد الإجراء يُمكّن أي مكون يتبع لأي تطبيق آخر أن ينفذ هذا الإجراء إذا كان مصمماً لذلك. على سبيل المثال، إذا اراد التطبيق عرض صفحة ويب معينة، يجب تشغيل فعالية (Activity) قادرة على عرض وتصفح الإنترنت، ويتم ذلك بتنفيذ الدالة startActivity() ويمرر للدالة هدف مضمن يحدد فيه إجراء باسم ACTION_VIEW وصفحة الويب المراد فتحها (لاحظ أنه لم يحدد اسم مكون محدد لعرض صفحة الويب). يقوم النظام تلقائياً باختيار الفعالية الملائمة لتشغيل الإجراء المطلوب (ACTION_VIEW)، وقد تكون الفعالية الملائمة تابعة لتطبيق آخر في نفس الجهاز.

عند إنشاء هدف صريح (Explicit Intent) بهدف تشغيل مكون ما فإن النظام يقوم مباشرة بتشغيل المكون المحدد في الهدف الصريح، أما عند إنشاء هدف مضمن (Implicit Intent) فإن النظام يبحث عن المكون المناسب لتنفيذ الإجراء المحدد في الهدف (Intent) وذلك في كل التطبيقات الموجودة في الجهاز. إذا توافق الهدف مع تطبيق معين، يقوم النظام بتشغيل التطبيق الموافق ويرسل له الهدف (Intent) والذي قد يحوي بيانات أخرى لإرسالها للتطبيق المطلوب.عند وجود أكثر من فعالية يمكنها تنفيذ الإجراء المطلوب يقوم النظام بعرض قائمة الفعاليات حتى يختار المستخدم الفعالية التي يريدها. شكل 1-4 يوضح إجراءات تشغيل فعالية (Activity B) من فعالية (Activity A) باستخدام هدف مضمن (Implicit Intent).



شكل 1-4: إنشاء فعالية باستخدام هدف مضمن: 1- من داخل الفعالية A يتم تنفيذ التعليمة startActivity() ويمرر إليها هدف مضمن يوضح الإجراء المراد تنفيذه. 2- يتقبل النظام الهدف ويبحث في كل التطبيقات في

الجهاز عن أي فعالية يمكنها تشغيل الإجراء المطلوب. فمثلاً، الفعالية B تحقق الغرض، لذلك يقوم النظام بتشغيلها ويرسل لها الهدف. 3- عند تشغيل الفعالية B يتم تنفيذ الدالة ()onCreate حيث يمكن خلالها استقبال الهدف الذي ارسله النظام والذي قد يحتوي على معلومات تلزم الفعالية B.[1]

مكونات الهدف (Intent)

الهدف (Intent) يحوي المعلومات اللازمة لتشغيل مكون ما وتشمل أهم هذه المعلومات ما يلي:

- المكون (Component): وهو يحدد المكون الذي يجب أن يستقبل الهدف (Intent). يجب تحديد اسم المكون فقط عند استخدام هدف صريح (Explicit Intent)، وهو ما يعني أن الهدف يجب أن يرسل فقط للمكون المحدد بالإسم. بدون تحديد اسم مكون يصبح الهدف مضمناً (Implicit Intent)، وهو ما يعني أن النظام هو من يحدد المكون الذي يمكنه استقبال الهدف بناءً على المعلومات الأخرى المضمنة في الهدف. لذلك إذا أردت تشغيل مكون محدد في تطبيقك فيجب عليك تحديد المكون من خلال هدف صريح (Explicit Intent) . يجب الملاحظة أن تحديد المكون يعني هنا الفئة (class) الخاصة بهذا المكون مثل الفعالية (Activity).
- الإجراء (Action): وهو جملة تحدد الإجراء المراد تنفيذه مثل عرض صفحة ويب أو الاتصال على رقم جوال وغيره. يتم تحديد الإجراء (Action) في حالة الهدف المضمن (Implicit Intent) فقط، حيث يترك للنظام مسؤولية إختيار المكون القادر على تنفيذ الإجراء. يمكن تحديد الإجراء الذي يستخدمه الهدف برمجياً إما بتمريره من خلال منشئ الباني constructor الخاص بالهدف أو بتنفيذ الدالة ()setAction. هناك بعض الإجراءات العامة التي يتعرف عليها النظام والتي يمكن استخدامها لأعراض عامة، ومن أهم هذه الإجراءات:
  - ACTION_VIEW: والذي يستخدم لعرض بيانات معينة للمستخدم. قد تكون هذه البيانات عبارة عن عنوان موقع ويب أو صورة يراد عرضها أو موقع على الخريطة وغيره.
  - ACTION_SEND: وهو يستخدم لمشاركة البيانات ارسالها كبريد إلكتروني أو تحميلها على شبكة إجتماعية.
- البيانات (Data): وهي عبارة عن كائن من نوع Uri يشير إلى البيانات المراد استخدامها. نوع البيانات المرسلة في الهدف (Intent) يعتمد على الإجراء المطلوب، فمثلا في حالة الإجراء (ACTION_VIEW) تكون البيانات على شكل عنوان الصفحة أو الصورة المراد عرضها.
- التصنيف (Category): وهو يحدد نوع المكون الذي يتلقى الهدف (Intent). في معظم الحالات لن تحتاج لتحديد التصنيف. من أنواع التصنيفات CATEGORY_LAUNCHER وهو يحدد الفعالية الرئيسية التي سيبدأ بها تشغيل التطبيق، حيث أن كل تطبيق له فعالية واحدة فقط من هذا التصنيف.
- الإضافات (Extras): وهو عبارة عن حافظة تستخدم لإرسال بيانات إضافية من خلال الهدف (Intent)، وترسل البيانات على شكل مفاتيح وقيم مقابلة (key-value pairs)، حيث يمكن الوصول لكل قيمة من خلال المفتاح المحدد لها. يتم إضافة البيانات للحافظة عن طريق الدوال من نوع ()putExtra. على سبيل المثال، اذا

[1] Intents and Intent Filters, Android Developer, http://developer.android.com/guide/components/intents-filters.html

اردت ارسال بريد إلكتروني من خلال التطبيق، يجب انشاء هدف وتحديد الإجراء له من نوع (ACTION_SEND). ثم يجب تحديد العنوان البريدي للمستقبل عن طريق الدالة putExtra وتستخدم المفتاح EXTRA_EMAIL كمفتاح لبيانات المستقبل، ويمكن أيضا تحديد موضوع الرسالة البريدية subject من خلال اضافة الموضوع بالمفتاح EXTRA_SUBJECT.

- Flag: وهو متغير يحدد كيفية إنشاء الفعالية وما المهمة التي ستتبع لها وكيف سيتم التعامل معها بعد الإنشاء. في الغالب لن تحتاج إلى تحديد قيم لهذا المتغير.

تشغيل الفعالية (Activity) باستخدام الهدف (Intent)

يمكن تشغيل فعالية عن طريق فعالية أخرى ما عن طريق تنفيذ الدالة startActivity() وتمرير هدف صريح (Explicit Intent) يحدد الفعالية المراد تشغيلها. الهدف (Intent) قد يحوي بالاضافة إلى الفعالية المراد تشغيلها بيانات أخرى مراد تمريرها للفعالية الجديدة.

الكود التالي يوضح تشغيل الفعالية SignInActivity باستخدام هدف صريح (Explicit Intent) وتمرير بيانات لها عن طريق حافظة الإضافات (Extras) الموجودة ضمن الهدف (Intent):

```
Intent intent =new Intent(getApplicationContext(),
SignInActivity.class);
intent.putExtras("username","Ahmed");
startActivity(intent);
```

لاحظ أن إضافة البيانات يتم في الحافظة Extras بطريقة المفتاح والقيمة (key-value)، حيث يستخدم المفتاح (key) لاحقاً لاسترجاع البيانات من الهدف (Intent). الفعالية التي تم تشغيلها باستطاعتها قراءة الهدف (Intent) عن طريق تنفيذ الدالة getIntent() ومن ثم استخراج البيانات المرسلة من خلال الهدف لمعالجتها.

في حالات أخرى قد تحتاج إلى تنفيذ إجراء معين (Action) مثل إرسال بريد إلكتروني أو رسالة نصية. في هذه الحالة قد لا يملك البرنامج أي فعالية محددة لتنفيذ الإجراء المطلوب، ويمكن استخدام فعاليات توفرها تطبيقات أخرى موجودة على الجهاز. يمكن تنفيذ ذلك باستخدام هدف مضمن (Implicit Intent) يحدد نوع الإجراء المراد تنفيذه حيث سيقوم النظام تلقائياً باختيار الفعالية المناسبة للإجراء المطلوب من أي تطبيق آخر على الجهاز. في حالة وجود أكثر من فعالية تدعم الإجراء المطلوب (مثل وجود أكثر من مستعرض لصفحات الانترنت)، يقوم النظام بطلب تدخل المستخدم لاختيار الفعالية التي يريدها لتنفيذ الإجراء المطلوب. على سبيل المثال، اذا أردت السماح لمستخدم بإرسال بريد إلكتروني من خلال تطبيق معين، يمكن تنفيذ ذلك بالكود التالي:

```
Intent intent =new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

لاحظ أنه تم إنشاء الهدف (Intent) باستخدام الإجراء المطلوب (ACTION_SEND) ولم يتم تحديد فعالية محددة لتشغيلها. الثابت EXTRA_EMAIL المضاف للهدف (Intent) يحدد المصفوفة recipientArray والتي تحوي العناوين البريدية المراد إرسال البريد إليها. هذه العناوين البريدية يتم إدارجها في خانة "إلى" (to) ضمن نموذج إرسال البريد.

ملاحظة هامة: عند إنشاء الفئة (class) الخاصة بالفعالية الجديدة تأكد أن الفعالية تم تعريفها في ملف الوثيقة (Manifest) وذلك بإضافة الخاصية activity داخل الخاصية application في ملف الوثيقة Manifest كما هو موضح:

```
<application>
…
  <activity
     android:name="ps.edu.ucas.example.NewActivity"
     android:label="@string/title_activity_details" >
  </activity>
…
</application>
```

حيث أن قيمة الخاصية android:name تشير إلى مسار الفئة (classpath) الخاص بالفعالية الجديدة.

تمرين عملي (1-4)

الوحدة الخامسة:

القوائم وشريط العمل

(Menus and Action Bar)

يتعلم الطالب في هذه الوحدة:

- ✓ أنواع القوائم المختلفة في تطبيق الأندرويد (قائمة الخيارات Options Menu، قائمة السياق Content Menu، والقائمة المنبثقة) واستخداماتها.
- ✓ إنشاء القوائم المختلفة برمجياً وربطها بالتطبيق
- ✓ تهيئة شريط العمل (Action Bar) والتحكم به برمجياً.
- ✓ إنشاء درج التصفح (Navigation Drawer) وربطه بالتطبيق

القوائم Menus من المكونات شائعة الاستخدام في تصميم واجهات التطبيقات. تتطرق هذه الوحدة إلى الأنواع المختلفة للقوائم وطريقة إنشائها.

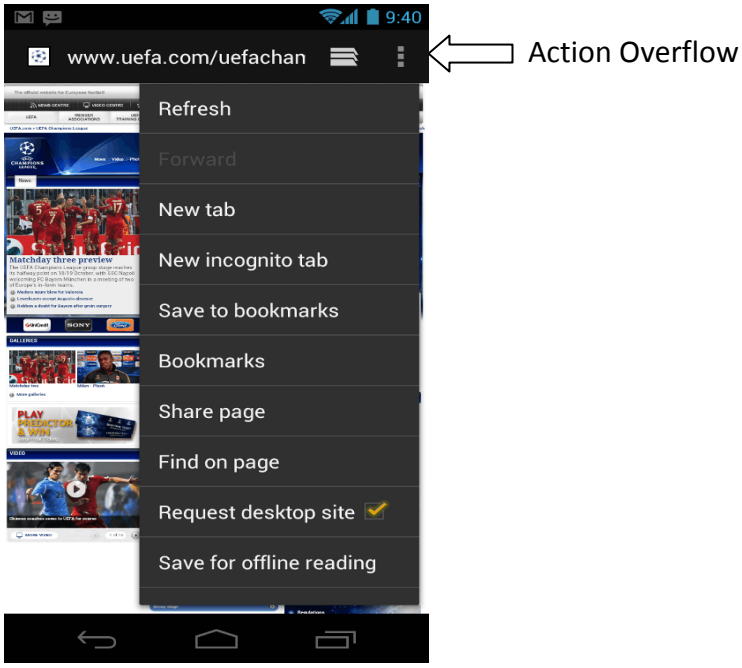
قائمة الخيارات (Options Menu) وشريط المهام (Action Bar)

تستخدم قائمة الخيارات Options Menu عادةً للوصول إلى الإجراءات العامة التي يتطلب الوصول إليها كثيراً أثناء استخدام التطبيق مثل إجراء البحث Search والإعدادات Settings. (ابتداءً من اصدار أندرويد 3.0 API level 11) يتم تضمين قائمة الخيارات ضمن شريط العمل (Action Bar)). يقوم النظام تلقائياً بإضافة عناصر القائمة الخيارات (Options Menu) إلى ملحق شريط العمل المسمى (Action Overflow) (أنظر شكل 5.1). لا تظهر عناصر القائمة تلقائياً، حيث يمكن عرضها بالنقر على اللإيقونة على يمين شريط العمل. يمكن أيضاً إظهار قائمة الخيارات بالنقر على زر "Menu" إذا كان الجهاز يوفر مثل هذا الزر.

لإنشاء قائمة اختيارات Options Menu لفعالية معينة (Activity)، يمكن في البداية تصميم محتويات القائمة خارجياً باستخدام XML. على الرغم من إمكانية بناء محتويات القائمة برمجياً، إلا أنه يفضل تصميم القائمة باستخدام XML وحفظها في مجلد المصادر. يمكن بعد ذلك إنشاء القائمة عند تشغيل الفعالية عن طريق الدالة ()Inflater.inflate. إنشاء القائمة في ملف XML يتيح إمكانية فصل تصميم القائمة عن عمل التطبيق. كما يتيح عمل إعدادات أو تصميمات مختلفة للقائمة بناءً على إعدادات الجهاز المختلفة مثل حجم شاشة العرض. كما ذكرنا في الوحدة الثالثة يمكن مواءمة التطبيق مع الإعدادات المختلفة عن طريق حفظ وترتيب الملفات الموجودة في مجلدات المصادر.

لتصميم قائمة خيارات جديدة، قم بإنشاء ملف XML جديد في المجلد /res/menu وقم ببناء القائمة داخله. الملف التالي my_menu.xml يوضح مثال لتصميم قائمة خيارات:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/ucas_homepage"
          android:icon="@drawable/ucas_logo"
          android:title="@string/ucas_homepage"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```



شكل 1-5: قائمة الخيارات (OptionsMenu)

كما هو موضح في المثال، تصميم القائمة كاملةً يكون ضمن الخاصية <menu> والتي تحتوي على عنصر أو أكثر من الخصائص <item> والتي تمثل عناصر القائمة. الخاصية <item> تستخدم لبناء عنصر القائمة (MenuItem).

من داخل الخاصية <item> يتم تعريف شكل عنصر القائمة (MenuItem) باستخدام الخصائص التالية:

- android:id : وهو يحدد رقم معرف (ID) خاص بعنصر القائمة (MenuItem). الرقم المعرف (ID) يمكّن التطبيق من معرفة عنصر القائمة (MenuItem) الذي تم النقر عليه، حيث كل عنصر قائمة له معرف خاص به.
- android:icon : وهو يضاف إختياريا لتحديد صورة الأيقونة الخاصة بعنصر القائمة. يجب أن تكون هذه الصورة ضمن مجلدات drawable.
- android:title: وهو يحدد النص المعروض في العنصر.
- android:showAsAction: وهو يحدد كيف ومتى يظهر عنصر القائمة ضمن شريط العمل (Action Bar). هناك قيم مختلفة يمكن تحديدها لهذه الخاصية مثل القيمة "ifRoom" وذلك لإظهار عنصر القائمة ضمن شريط العمل إذا كان هناك متسع به، وكذلك القيمة "never" لعدم إظهاره ضمن شريط العمل، والقيمة "always" لإظهاره دائماً ضمن شريط العمل

تمثل هذه أهم الخصائص التي ينصح باستخدامها، ولكن هناك العديد من الخصائص الأخرى التي لن نتطرق لها في هذا الكتاب.

يمكن أيضاً إضافة قائمة فرعية (submenu) داخل أي عنصر في القائمة (Menu) وذلك بإضافة الخاصية <menu> ضمن الخاصية <item>. القوائم الفرعية (submenus) مفيدة في حالة تعدد وتفرع الإجراءات التي تنفذ من خلال القائمة أو لترتيب عناصر القائمة ضمن مجموعات. المثال التالي يوضح إضافة قائمة فرعية إلى عنصر قائمة:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

بعد تصميم قائمة الخيارات (Options Menu) يتم استخدامها في الفعالية (Activity) عن طريق الدالة MenuInflater.inflate() والتي نقوم بتحويل التصميم في ملف XML إلى عنصر واجهة يتم الوصول إليه برمجياً. يتم كتابة الدالة onCreateOptionsMenu()، وهي أحدى الدوال الموجودة ضمن الفعالية (Activity)

والتي يتم تنفيذها تلقائياً عند إنشاء الفعالية، وذلك لبناء القائمة من ملف XML باستخدام الكائن MenuInflater كما هو موضح:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.my_menu, menu);
    return true;
}
```

معالجة أحداث القائمة (Handling Click Events)

عندما يختار المستخدم عنصر من قائمة الخيارات (Options Menu) (بما فيها العناصر في شريط العمل Action Bar)، يقوم النظام بتنفيذ الدالة ()onOptionsItemSelected والتي يمرر لها عنصر القائمة MenuItem الذي تم اختياره (أنظر الكود بالأسفل). يمكن تمييز هذا العنصر عن طريق تنفيذ الدالة ()getItemId والتي ترجع المعرف ID الخاص به والمنشأ باستخدام الخاصية android:id في ملف XML الخاص بتصميم القائمة. بعد معالجة الحدث يتم إرجاع القيمة true. إذا لم يتم معالجة الحدث يجب تنفيذ الدالة ()onOptionsItemSelected الموجودة في الفئة الأم (superclass).

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

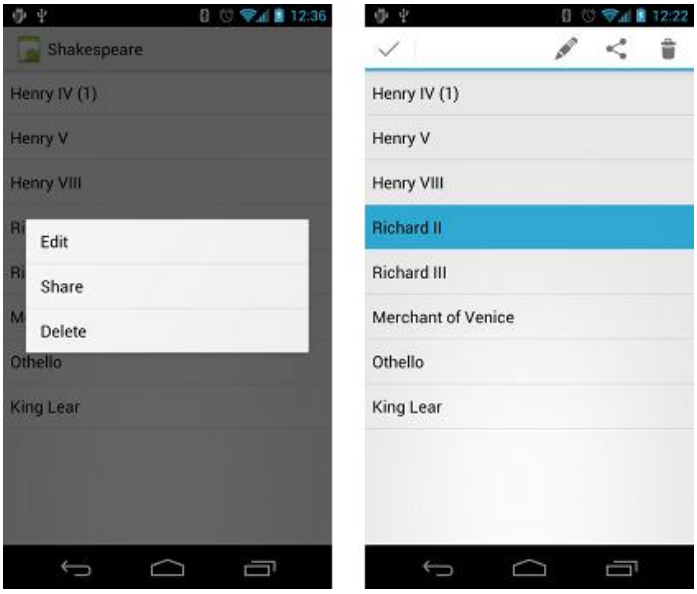
يوفر نظام أندرويد أيضاً إمكانية تعريف دالة لمعالجة حدث النقر click من خلال ملف XML الخاص بتصميم القائمة باستخدام الخاصية android:onClick. القيمة التي تأخذها هذه الخاصية هي اسم الدالة التي تعالج الحدث والتي يجب تكون عامة public ويمرر لها متغير من نوع MenuItem.

كما ذكرنا مسبقاً، يتم إنشاء القائمة Menu من خلال الكود الموجود في الدالة ()onCreateOptionsMenu. إذا أردت تعديل محتويات القائمة بالإضافة أو التعديل أو الحذف أثناء عمل الفعالية (Activity)، يمكن القيام بذلك من خلال الدالة ()onPrepareOptionsMenu. هذه الدالة يمرر إليها كائن من النوع Menu والخاص بالقائمة التي تم انشاؤها مسبقاً في الدالة ()onCreateOptionsMenu، حيث يمكن التعديل عليها بالإضافة أو الحذف.

قائمة السياق (Context Menu)

قائمة السياق Context Menu يتم من خلالها تنفيذ الإجراءات التي تؤثر على عنصر محدد في واجهة المستخدم. يمكنك ربط قائمة السياق Context Menu بأي عنصر واجهة View، ولكن غالبا ما تستخدم لتنفيذ اجراءات على العناصر في القائمة (ListView) أو (GridView)، أو غيرها من عناصر عرض المجموعات التي يمكن للمستخدم تنفيذ إجراءات مباشرة على كل بند فيها. هناك طريقتان لعرض قائمة السياق (Context Menu)، كما هو موضح بشكل 2-5، وهما:

- Floating Context Menu: حيث تظهر كقائمة عائمة عندما يقوم المستخدم بنقرة وطويل على عنصر واجهة View. باستخدام هذا الوضع (Floating Context Menu)، يمكن تنفيذ الإجراءات على عنصر واحد فقط في الوقت الواحد.
- Contextual Action Mode: وفيه يتم عرض عناصر القائمة (Menu)،التي تؤثر على العنصر الذي تم اختياره ،ضمن شريط العمل (Action Bar) في الجزء العلوي من الشاشة. عندما يكون هذا الوضع مفعلاً، يمكن للمستخدمين تنفيذ إجراء على عناصر متعددة في وقت واحد (إذا كان التطبيق الخاص بك يسمح بذلك) (لن يتم التطرق للقوائم من نوع Contextual Action Mode في هذا الكتاب).



شكل 2-5: قائمة سياق عائمة (Floating Context Menu) على اليسار، وشريط عمل السياق (Contextual Action Bar) على اليمين.

إنشاء قائمة سياق عائمة (Floating Context Menu)

يتم إنشاء قائمة السياق العائمة (Floating Context Menu) باستخدام الخطوات التالية:

1. كما ذكرنا مسبقاً، قائمة السياق العائمة ترتبط بعنصر واجهة (View) محدد وتظهر عند النقر عليه. عنصر الواجهة (View) المراد ربط قائمة السياق (Context Menu) به يجب تسجيله بتنفيذ الدالة registerForContextMenu() ضمن الفعالية (Activity) وتمرير العنصر (View) لها. إذا كانت الفعالية (Activity) تستخدم قائمة عرض (ListView) وأردت استخدام نفس قائمة السياق Context Menu لكل القائمة (ListView)، يجب في هذه الحالة تسجيل القائمة (ListView) بتمريرها للدالة registerForContextMenu().
2. يتم كتابة كود في الدالة onCreateContextMenu() والموجودة ضمن الفعالية (Activity). يتم من خلال هذه الدالة إنشاء القائمة من مصدرها في ملف XML وذلك باستخدام كائن من نوع MenuInflater كما هو موضح في المثال التالي:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}
```

لاحظ أن الدالة onCreateContextMenu() يمرر لها ،بالإضافة للقائمة (ContextMenu)، العنصر View والذي تم اختياره لتفعيل قائمة السياق (Context Menu) وكائن من النوع ContextMenu.ContextMenuInfo والذي يحتوي على معلومات إضافية خاصة بالعنصر الذي تم اختياره. هذه المعلومات يتم استخدامها اذا كان هناك عناصر Views متعددة وكل منها مرتبط بقائمة سياق Context Menu مختلفة. في هذه الحالة تفيد هذه المعلومات في تحديد القائمة التي يجب إنشاؤها لكل عنصر عرض View.

3. كتابة كود الدالة onContextItemSelected() ضمن الفعالية (Activity) وذلك لتحديد الإجراء المراد تنفيذه عند النقر على أي من عناصر القائمة Context Menu. المثال التالي يوضح ذلك:

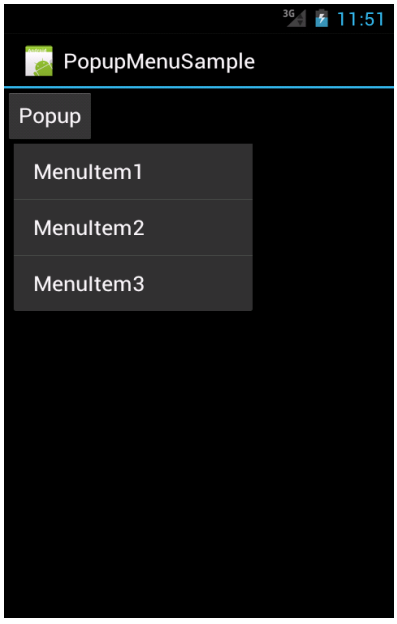
```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo)
item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
```

```
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

القائمة المنبثقة (Popup Menu)

القائمة المنبثقة (Popup Menu) (أنظر شكل 5.3) هي قائمة ملحقة بعنصر واجهة View وتظهر أسفله (أو أعلاه إلى لم تتوفر مساحة لعرضها) كما بشكل. القائمة المنبقة (Popup Menu) يمكن أن تستخدم في الحالات التالية:

1. إظهار قائمة من الإجراءات المتعلقة بمحتوى معين. على سبيل المثال، عند النقر على زر في شريط العمل (Action Bar) يتم اظهار قائمة منبثقة تتضمن مجموعة من الإجراءات.
2. إظهار قائمة من الإجراءات التفصيلية التي تتفرع من إجراء أساسي. فمثلاً، عند النقر على زر "إرسال"، تظهر قائمة منبثقة من إجراءات "إرسال" أخرى مثل إرسال إلى الجميع، إرسال إلى صديق، إرسال إلى مجموعة، وهكذا.



شكل 3-5: القائمة المنبثقة (Popup Menu)

لاحظ أن القائمة المنبقة متوفرة في ٌإصدار أندرويد API Level 11 وما بعده.

إظهار القائمة المنبثقة (Popup Menu) يتم بالخطوات التالية:

1. أنشئ كائن من نوع PopupMenu بحيث يمرر إلى الباني constructor الخاص بها الكائن Context الخاص بالتطبيق بالإضافة إلى عنصر الواجهة view التي يراد إلحاق القائمة به كما هو موضح:
2. استخدام كائن من نوع MenuInflater لإنشاء القائمة باستخدام ملف التصميم XML الخاص بالقائمة.
3. إظهار القائمة بتنفيذ الدالة Poupmenu.show().

على سبيل المثال، الكود بالأسفل يوضح التصميم الخاص بزر Button والذي عند النقر عليه يتم تنفيذ الدالة showPopup والمسؤولة عن إظهار قائمة منبثقة (Popup Menu).

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_overflow_holo_dark"

android:contentDescription="@string/descr_overflow_button"
    android:onClick="showPopup" />
```

الكود التالي يوضح الدالة showPopup() داخل الفعالية (Activity):

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.actions, popup.getMenu());
    popup.show();
}
```

ملاحظة: إبتداءً من الإصدار API Level 14 يمكن استخدام الدالة PopupMenu.inflate() بدلاً من استخدام MenuInflater.inflate().

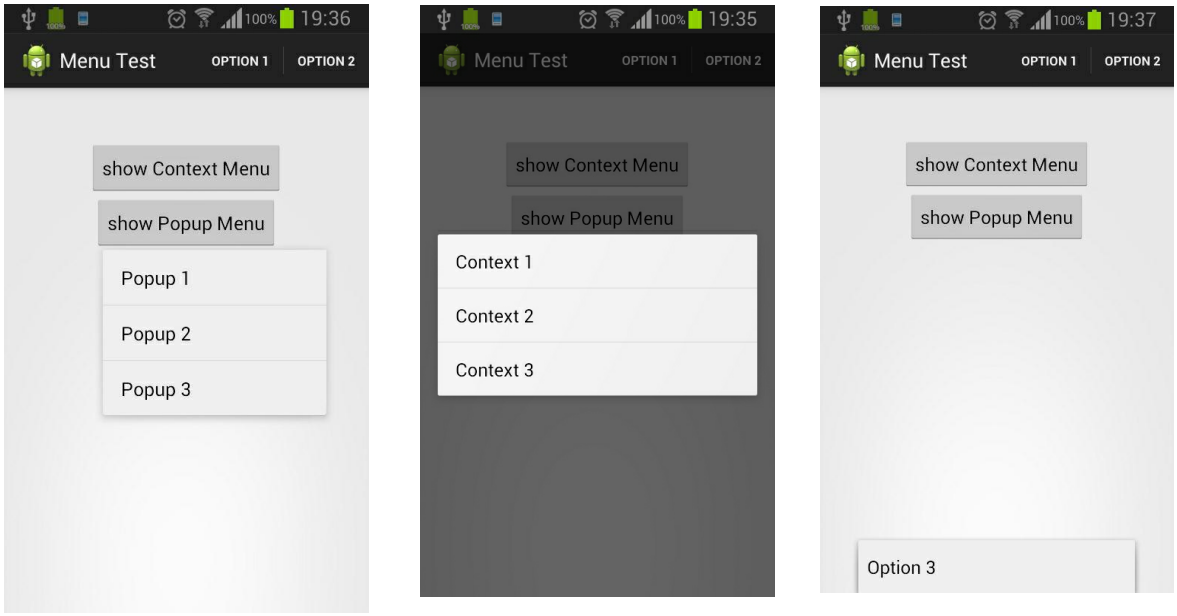
لتنفيذ إجراء معين عند النقر على أي عنصر في القائمة المنبثقة Popup يجب إنشاء مستمع (Listener) من نوع MenuPopupMenu.OnMenuItemClickListener وتسجيله باستخدام الدالة PopupMenu.setOnMenuItemClickListener() ومن ثم كتابة كود الإجراء المطلوب في الدالة onMenuItemClick(). الكود التالي يوضح هذه الخطوة:

```
public void showMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);

    popup.setOnMenuItemClickListener(new
OnMenuItemClickListener(){

     @Override
public boolean onMenuItemClick(MenuItem item) {
      switch (item.getItemId()) {
          case R.id.archive:
              archive(item);
              return true;
          case R.id.delete:
              delete(item);
              return true;
          default:
              return false;
      }
    }

  });
  popup.inflate(R.menu.actions);
  popup.show();
}
```

تمرين عملي (1-5)

يهدف هذا التمرين إلى التدريب على إنشاء أنواع القوائم (Menus) المختلفة التي تم تفصيلها سابقاً في هذه الوحدة. واجهة التطبيق والقوائم التي يتم عرضها موضحة في شكل 4-5. يستخدم التطبيق قائمة خيارات (Options Menu) تظهر في شريط العمل (option 1, option 2,..) بينما تظهر بقية العناصر عند النقر على أيقونة Action Overflow أو النقر على زر "Menu" في الجهاز.

تحتوي واجهة التطبيق على زرين: "Show Context Menu"، "Show Popup Menu" عند النقر عليهما تظهر قائمة السياق (Context Menu) و القائمة المنبثقة (Popup Menu) كما هو موضح في شكل 4-5. عند النقر على عنصر في أي من القوائم الثلاث يتم طباعة رسالة باستخدام Toast تظهر اسم العنصر.



شكل 4-5: واجهة التطبيق الخاص بتمرين 1-5: قائمة الخيارات Options Menu تظهر في شريط العمل Action Bar (يمين)، قائمة السياق Context Menu (وسط)، القائمة المنبثقة Popup Menu (يسار)

تصميم الواجهة الرئيسية (MainActivity) موضح فيما يلي:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.menutest.MainActivity" >

    <Button
        android:id="@+id/contextBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
```

```
        android:layout_centerHorizontal="true"
        android:text="show Context Menu"
        android:layout_marginTop="30dp" />

    <Button
        android:id="@+id/popupBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/contextBtn"
        android:layout_centerHorizontal="true"
        android:text="show Popup Menu"
        android:onClick="showPopup" />

</RelativeLayout>
```

ملفات XML الخاصة بالقوائم الثلاثة موضحة فيما يلي:

أولاً: قائمة الخيارات (Options Menu)

```
//options_menu.xml
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android"
>
    <item android:id="@+id/option1" android:title="Option
1" android:showAsAction="ifRoom"></item>
    <item android:id="@+id/option2" android:title="Option
2" android:showAsAction="ifRoom"></item>
    <item android:id="@+id/option3" android:title="Option
3" android:showAsAction="ifRoom"></item>
</menu>
```

ثانياً: قائمة السياق (Context Menu)

```
// context_menu.xml
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android"
>
    <item android:id="@+id/context1" android:title="Context
1"></item>
    <item android:id="@+id/context2" android:title="Context
2"></item>
```

```
    <item android:id="@+id/context3" android:title="Context
3"></item>
</menu>
```

ثالثاً: القائمة المنبثقة (Popup Menu)

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android"
>
    <item android:id="@+id/popup1" android:title="Popup
1"></item>
    <item android:id="@+id/popup2" android:title="Popup
2"></item>
    <item android:id="@+id/popup3" android:title="Popup
3"></item>
</menu>
```

الكود التالي يوضح الفعالية (MainActivity):

```
1:  public class MainActivity extends Activity {
2:
3:   @Override
4:   protected void onCreate(Bundle savedInstanceState) {
5:        super.onCreate(savedInstanceState);
6:        setContentView(R.layout.activity_main);
7:
8:  // Linking the button contextBtn with the context menu
9:        Button contextBtn = (Button)
10: this.findViewById(R.id.contextBtn);
11:       this.registerForContextMenu(contextBtn);
12:  }
13:
14:  @Override
15:  public boolean onCreateOptionsMenu(Menu menu) {
16:
17:       getMenuInflater().inflate(R.menu.options_menu,
18:  menu);
19:       return true;
20:  }
```

```
21:
22:  @Override
23:  public boolean onOptionsItemSelected(MenuItem item) {
24:     switch (item.getItemId()) {
25:       case R.id.option1:
26:            Toast.makeText(this, "Option 1 selected",
27:  Toast.LENGTH_LONG).show();
28:            return true;
29:       case R.id.option2:
30:           Toast.makeText(this, "Option 2 selected",
31:  Toast.LENGTH_LONG).show();
32:            return true;
33:       case R.id.option3:
34:           Toast.makeText(this, "Option 3 selected",
35:  Toast.LENGTH_LONG).show();
36:            return true;
37:       default:
38:           return super.onOptionsItemSelected(item);
39:     }
40:  }
41:
42:  @Override
43:  public void onCreateContextMenu(ContextMenu menu, View
44:  v, ContextMenuInfo menuInfo) {
45:       super.onCreateContextMenu(menu, v, menuInfo);
46:  this.getMenuInflater().inflate(R.menu.context_menu,
47:  menu);
48:  }
49:
50:  @Override
51:  public boolean onContextItemSelected(MenuItem item) {
52:       switch (item.getItemId()) {
53:         case R.id.context1:
54:              Toast.makeText(this, "Context 1 selected",
55:              Toast.LENGTH_LONG).show();
56:              return true;
57:         case R.id.context2:
58:             Toast.makeText(this, "Context 2 selected",
59:  Toast.LENGTH_LONG).show();
60:             return true;
```

```
61:         case R.id.context3:
62:             Toast.makeText(this, "Context 3 selected",
63: Toast.LENGTH_LONG).show();
64:             return true;
65:         default:
66:             return super.onContextItemSelected(item);
67:          }
68:  }
69:
70:  public void showPopup(View view) {
71:      PopupMenu popup = new PopupMenu(this, view);
72:      popup.setOnMenuItemClickListener(new
73:  OnMenuItemClickListener(){
74:
75:            @Override
76:            public boolean onMenuItemClick(MenuItem
77:  item) {
78:                    switch (item.getItemId()) {
79:                  case R.id.popup1:
80:
81:  Toast.makeText(getApplicationContext(), "Popup 1
82:  selected", Toast.LENGTH_LONG).show();
83:                       return true;
84:                  case R.id.popup2:
85:
86:  Toast.makeText(getApplicationContext(), "Popup 2
87:  selected", Toast.LENGTH_LONG).show();
88:                       return true;
89:                  case R.id.popup3:
90:
91:  Toast.makeText(getApplicationContext(), "Popup 3
92:  selected", Toast.LENGTH_LONG).show();
93:                       return true;
94:                  default:
95:                       return false;
96:                    }
97:            }
98:      });
99:      MenuInflater inflater = popup.getMenuInflater();
100:     inflater.inflate(R.menu.popup_menu,
```

```
101:  popup.getMenu());
102:     popup.show();
103: }
104:}
```

لاحظ أنه في الدالة ()onCreate عند تشغيل الفعالية، يتم الوصول للزر "contextButton" وربطه بقائمة السياق عند طريقة الدالة ()registerForContextMenu (أنظر سطر رقم 11).

قائمة الخيارات (Options Menu) يتم إنشاؤها ضمن الدالة ()onCreateOptionsMenu (أنظر الكود من سطر 15 إلى 20) ويتم معالجة الأحداث المرتبطة بها ضمن الدالة ()onOptionsItemSelected (أنظر الكود من سطر 23 إلى 40).

قائمة السياق (Context Menu) يتم إنشاؤها ضمن الدالة ()onCreateContextMenu (أنظر الكود من سطر 43 إلى 48) ويتم معالجة الأحداث المرتبطة بها ضمن الدالة ()onContextItemSelected (أنظر الكود من سطر 51 إلى 68).

القائمة المنبثقة (Popup Menu) يتم إنشاؤها ضمن الدالة ()showPopup (أنظر الكود من سطر 70 إلى 103) والتي تنفذ عن النقر على الزر المخصص.

درج التصفح (Navigation Drawer)

درج التصفح (Navigation Drawer) هو لائحة تشتمل على عناصر للتحكم بالتطبيق، مثل الأزرار أو القائمة (ListView)، علي يمين أو يسار الشاشة. تكون هذه اللائحة مخفية تلقائياً ويتم إظهارها عند السحب (swipe) باتجاه العرض (أنظر شكل 5-5). درج التصفح من الأدوات شائعة الاستخدام في تطبيقات أندرويد.

لإنشاء درج تصفح (Navigation Drawer) ضمن تطبيقك، يجب تصميم الواجهة باستخدام تصميم من نوع DrawerLayout. يجب أن يحتوي التصميم على جزئين:

1. المحتوى الأساسي (Main Content View): هذا الجزء يمثل المحتوى الأساسي للواجهة والذي يظل ثابتاً، ويجب أن يأتي في الترتيب الأول داخل ملف التصميم. يمكن أن يكون هذا الجزء عبارة عن تصميم (Layout) داخل التصميم (DrawerLayout) يحوي داخله مجموعة من عناصر العرض .
2. محتوى الدرج (Drawer Content) الجزء الثاني من التصميم DrawerLayout يمثل محتوى درج التصفح والذي يكون مخفياً ويظهر عند السحب فقط. يمكن أن يكون محتوى درج التصفح عبارة عن عنصر عرض وحيد (مثل ListView) أو مجموعة من العناصر يجمعها تصميم واحد (Layout) . محتوى الدرج يجب أن يستخدم خاصية XML التالية: android:layout_gravity والتي تأخذ القيمة "start" أو "end". هذه الخاصية تعني أن محتوى العرض يقع على يسار الشاشة (”start“) أو يمينها (”end“). كما يجب أن يستخدم محتوى الدرج الخاصية android:layout_width لتحديد عرض درج الحوار عند عرضه. لاحظ أن درج التصفح يجب أن لايخفي المحتوى الأساسي كاملاً عند عرضه.

شكل 5-5: درج التصفح (Navigation Drawer) [1]

المثال التالي يوضح تصميم لواجهة بسيطة: لاحظ أن تصميم الواجهة يستخدم DrawerLayout. بداخله تم استخدام تصميم من نوع RelativeLayout (أو أي تصميم آخر) والذي يجب أن تحتوي العناصر الأساسية الثابتة للواجهة (Main Content View). الجزء الثاني من التصميم DrawerLayout هو قائمة العرض (ListView) والتي تمثل محتويات درج التصفح. لاحظ أن القائمة تستخدم الخاصية android:layout_gravity.

```
<android.support.v4.widget.DrawerLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@android:style/Theme.WithActionBar"
    >
     <RelativeLayout >
     <!—Place main UI components here -->
     </RelativeLayout>
```

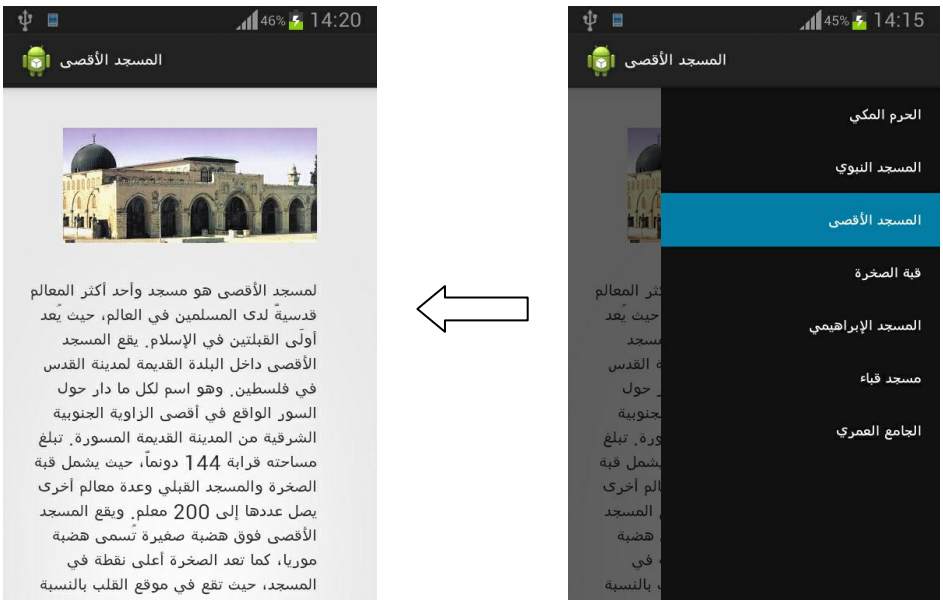
[1] Navigation Drawer, Android Developers, https://developer.android.com/design/patterns/navigation-drawer.html

```
    <ListView
        android:id="@+id/left_drawer"
         android:layout_gravity="start"
        android:layout_width="250dp"
        android:layout_height="match_parent"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

ملاحظة: يمكن إدراج المحتوى الرئيسي (Main Content View) ضمن تصميم خارجي يسمى بالقطعة (Fragment) ومن ثم إدراج القطعة (Fragment) داخل التصميم DrawerLayout. سيتم التطرق للقطع (Fragments) في الوحدة التاسعة.

تمرين عملي (2-5)

في هذا التمرين سنقوم ببناء تطبيق معلوماتي عن المساجد الشهيرة في فلسطين وواجهته موضحة في شكل 6-5. يستخدم التطبيق درج التصفح (Navigation Drawer) لعرض قائمة بالمساجد، وعند النقر على اسم مسجد تظهر تفاصيله ضمن الواجهة الرئيسية. درج التصفح يكون مخفياً، ويتم عرضه عن طريق السحب من اليمين لليسار.



شكل 6-5: واجهة التطبيق الخاص بتمرين 2-5.

نقوم في البداية بإنشاء مصفوفة نصوص string-array باسم mosques_array ضمن المصادر تحتوي أسماء المساجد لعرضها ضمن قائمة (ListView) داخل درج التصفح (Navigation Drawer). سيتم لاحقاً الوصول لهذه المصفوفة برمجياً لتعبئة القائمة (ListView).

```
<resources>
    <string name="app_name">Navigation Drawer
Example</string>
    <string-array name="mosques_array">
        <item>المكي الحرم</item>
        <item>النبوي المسجد</item>
        <item>الأقصى المسجد</item>
        <item>الصخرة قبة</item>
        <item>الإبراهيمي المسجد</item>
        <item>قباء مسجد</item>
        <item>العمري الجامع</item>
    </string-array>
</resources>
```

فيما يلي نعرض تصميم الواجهة الرئيسية والتي تتكون من المحتوى الرئيسيي (تفاصيل المسجد) و درج التصفح (Navigation Drawer):

```
//activity_main.xml
<android.support.v4.widget.DrawerLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@android:style/Theme.WithActionBar"
    >
     <ScrollView android:layout_width="fill_parent"
android:layout_height="fill_parent">
          <RelativeLayout
              android:layout_width="match_parent"
              android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin">
              <ImageView
                  android:id="@+id/imageView1"
```

```
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentTop="true"
                    android:layout_centerHorizontal="true"
                    android:layout_marginTop="20dp"
                    android:src="@drawable/ic_launcher" />
                <TextView
                    android:id="@+id/textView1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

android:layout_alignRight="@+id/imageView1"
                    android:layout_below="@+id/imageView1"

android:layout_gravity="center_vertical|right"
                    android:textSize="20sp"
                    android:layout_marginTop="30dp"
                    android:text="TextView" />

            </RelativeLayout>
     </ScrollView>
    <ListView
        android:id="@+id/right_drawer"
         android:layout_gravity="end"
        android:layout_width="250dp"
        android:layout_height="match_parent"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

لاحظ أن التصميم السابق (DrawerLayout) مكون من جزئين: الجزء الأول هو التصميم ScrollView ويحتوي داخله على تصميم RelativeView يضم عناصر العرض ImageView و TextView. العنصر ImageView يستخدم لعرض صورة المسجد بينما العنصر TextView يعرض معلومات عن المسجد. الجزء الثاني هو قائمة العرض (ListView) والتي تستخدم الخاصية "android:layout_gravity="end وذلك حتى يتم اخفاؤها في يمين الشاشة.

الكود التالي خاص بالفعالية MainActivity والتي يتم من خلالها إنشاء الواجهة ودرج التصفح ومعالجة الأحداث المختلفة:

```
1:  public class MainActivity extends Activity {
2:      private DrawerLayout mDrawerLayout;
3:      private ListView mDrawerList;
4:      private ImageView mosqueImage;
5:      private TextView mosqueDetails;
6:      private String[] mosqueNames;
7:
8:      @Override
9:      protected void onCreate(Bundle savedInstanceState)
10:    {
11:        super.onCreate(savedInstanceState);
12:        setContentView(R.layout.activity_main);
13:
14:        mosqueNames =
15:  getResources().getStringArray(R.array.mosques_array);
16:        mDrawerLayout = (DrawerLayout)
17:  findViewById(R.id.drawer_layout);
18:        mosqueImage = (ImageView)
19:  findViewById(R.id.imageView1);
20:        mosqueDetails = (TextView)
21:  findViewById(R.id.textView1);
22:        mDrawerList = (ListView)
23:  findViewById(R.id.right_drawer);
24:        mDrawerList.setAdapter(new
25:  ArrayAdapter<String>(this, R.layout.drawer_list_item,
26:  mosqueNames));
27:        mDrawerList.setOnItemClickListener(new
28:  ListView.OnItemClickListener() {
29:            @Override
30:            public void onItemClick(AdapterView<?>
31:  parent, View view, int position, long id) {
32:     mDrawerList.setItemChecked(position, true);
33:     String mosqueName = mosqueNames[position];
34:     setTitle(mosqueName);
35:     if(mosqueName.equalsIgnoreCase("المسجد الأقصى")){
36:       mosqueImage.setImageResource(R.drawable.aqsa);
37:       mosqueDetails.setText("...");
```

```
38:     }else if(mosqueName.equalsIgnoreCase("قبة الصخرة")){
39:        mosqueImage.setImageResource(R.drawable.dome);
40:        mosqueDetails.setText("...");
41:     }
42:     mDrawerLayout.closeDrawer(mDrawerList);
43:   }
44:});
45:         ActionBarDrawerToggle mDrawerToggle = new
46:  ActionBarDrawerToggle(
47:                 this,
48:                 mDrawerLayout,
49:                 R.drawable.ic_drawer,
50:                 R.string.drawer_open
51:                 ) {
52:     public void onDrawerClosed(View view) {}
53:
54:     public void onDrawerOpened(View drawerView){}
55:    };
56:  mDrawerLayout.setDrawerListener(mDrawerToggle);
57:     }
58:}
```

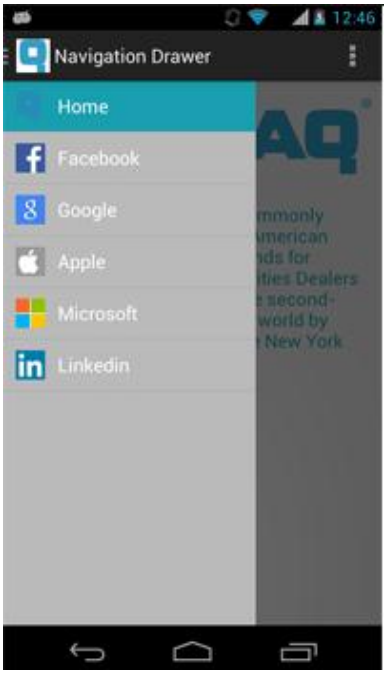
في الدالة onCreate() يتم الوصول لمصفوفة الأسماء mosques_array عن طريق الكود التالي:

mosqueNames = getResources().getStringArray(R.array.mosques_array); (أنظر سطر رقم 14). ومن ثم يتم تعبئة القائمة mDrawerList عن طريق استخدام ArrayAdapter (أنظر سطر رقم 24). بعد ذلك يتم الاستماع لحدث النقر على عناصر القائمة باستخدام الدالة setOnItemClickListener() وكتابة الدالة onItemClicked() (أنظر الكود من سطر 27 إلى 30). الإجراء الذي يتم تنفيذه عند النقر على القائمة هو تغيير عنوان الفعالية بتنفيذ الدالة setTitle() وتغيير محتوى ImageView و TextView لعرض معلومات عن المسجد الذي تم اختياره (أنظر الكود من سطر 32 إلى 41). كذلك يتم إخفاء درج التصفح (Navigation Drawer) عن طريق الدالة DrawerLayout.closeDrawer(); (سطر رقم 42).

يمكن كذلك الاستماع لأحداث فتح وإغلاق درج التصفح (Navigation Drawer) عن طريق الدالة DrawerLayout.setDrawerListener() والتي يمرر لها كائن من نوع ActionBarDrawerToggle. الإجراء المراد تنفيذه عن إغلاق أو فتح درج التصفح يتم كتابته في الدالتين onDrawerClose() و onDrawerOpened() على الترتيب. الكود في الأعلى (من سطر رقم 45 إلى 56) يوضح هذه الخطوة بالرغم من أنه لم يتم استخدامها لتنفيذ أي إجراء في هذا التطبيق.

أسئلة على الوحدة الخامسة

1. ما الفرق بين قائمة السياق (Context Menu) والقائمة المنبثقة (Popup Menu)؟ ومتى تستخدم كل منهما؟
2. قم بتعديل تمرين 4-3 والذي يعرض قائمة (ListView) تضم أسماء وصور مجموعة من الأصدقاء. عند الضغط لفترة على اسم صديق في القائمة يتم عرض قائمة سياق (Context Menu) تضمن زر حذف "Delete". عند النقر على زر الحذف يتم حذف السطر الخاص بالصديق من القائمة (ListView).
3. قم ببناء التطبيق الموضح بالشكل والذي يستخدم عنصر عرض من نوع WebView كمحتوى رئيسي ثابت بالإضافة إلى درج تصفح (Navigation Drawer) يضم قائمة بأسماء بعض المواقع الالكترونية الشهيرة مثل Facebook, Google, Apple وغيرها. درج التصفح يظهر عند السحب من اليسار لليمين، و عند النقر على أي منها، يتم إخفاء درج التصفح ومن ثم فتح الموقع في عنصر العرض WebView.



الوحدة السادسة:

استدامة البيانات في نظام أندرويد

(Data Persistence in Android)

يتعلم الطالب في هذه الوحدة:

- ✓ الطرق المختلفة لحفظ البيانات الخاصة بتطبيقات أندرويد ومتى يجب استخدام كل منها.
- ✓ إنشاء تطبيقات ترتبط بقواعد البيانات.
- ✓ تنفيذ إجراءات مختلفة على قواعد البيانات وعرض النتائج في واجهة المستخدم.

لدراسة هذه الوحدة لابد من الإلمام بمبادئ التعامل مع قواعد البيانات تعليمات SQL المختلفة. كذلك لابد من الإلمام بأساسيات القراءة والكتابة للملفات بلغة جافا.

يوفر نظام أندرويد أكثر من طريقة لحفظ البيانات الخاصة بالتطبيقات. أهم هذه الطرق تشمل ما يلي:

- التفضيلات المشتركة (Shared Preferences): وتشمل حفظ بيانات أساسية بمفاتيح محددة (key-value pairs).
- الذاكرة الداخلية (Internal Storage) وفيها يتم تخزين البيانات في الذاكرة الداخلية للجهاز.
- الذاكرة الخارجية (External Storage) وفيها يتم تخزين البيانات في ذاكرة خارجية مشتركة بين التطبيقات.
- قواعد البيانات (SQLite Databases): وفيها يتم تخزين البيانات في قواعد بيانات خاصة بالتطبيق.

استخدام أي من هذه الطرق يعتمد على احتياجات التطبيق مثل ما إذا كانت البيانات المراد حفظها خاصة بالتطبيق أو يمكن الوصول إليها من تطبيقات أخرى، وكذلك المساحة المطلوبة لتخزين البيانات. سيتم فيما يلي تفصيل كل من هذه الطرق:

التفضيلات المشتركة (Shared Preferences)

التفضيلات المشتركة (SharedPreferences) هي مكون يوفر حفظ واسترجاع بيانات أساسية بسيطة. كل معلومة يتم حفظها يجب أن تكون على شكل مفتاح وقيمة (key-value) بحيث يحدد المفتاح (key) اسم مميز ليتم استعادة القيمة (value) فيما بعد عن طريقه. يمكن استخدام التفضيلات المشتركة لحفظ بيانات من الأنواع البدائية (primitive types) وتشمل: booleans, floats, ints, longs, and Strings. يتم الاحتفاظ بالبيانات المخزنة من تطبيق ما في التفضيلات المشتركة حتى بعد إغلاق هذا التطبيق.

للوصول إلى الكائن الخاص بالتفضيلات المشتركة واستخدامه، يمكن استخدام واحدة من إحدى دالتين:

- getSharedPreferences(): استخدم هذه الدالة إذا أردت انشاء أكثر من ملف لحفظ البيانات. لإنشاء ملف تفضيلات جديد يتم تمرير اسم جديد من خلال الدالة getSharedPreferences().
- getPreferences(): في هذه الحالة يتم استخدام ملف واحد لكل فعالية لحفظ التفضيلات، ولا يتم تمرير أي اسم لهذه الدالة.

ولكتابة بيانات إلى التفضيلات المشتركة يجب القيام بالخطوات التالية:

1. تنفيذ الدالة edit() والتي تقوم بإرجاع كائن من نوع SharedPreferences.Editor والذي يتم من خلاله الكتابة.
2. أضف البيانات عن طريق تنفيذ مهام مثل: putBoolean(), putString(), putInt().
3. إعتماد تنفيذ عملية حفظ البيات المدخلة عن طريق تنفيذ الدالة commit().

المثال التالي يوضح كيفية حفظ متغير باسم mSilentMode من نوع boolean في التفضيلات المشتركة قبل إنهاء الفعالية مباشرة، ثم يتم استرجاع هذه القيمة عند تشغيل الفعالية مرة أخرى:

```
public class Example extends Activity{
    public static final String PREFS_NAME ="MyPrefsFile";
    private Boolean mSilentMode;

    @Override
    protectedvoid onCreate(Bundle state){
       super.onCreate(state);
       ...

       // Restore preferences
       SharedPreferences settings =
getSharedPreferences(PREFS_NAME,0);
       mSilentMode =
settings.getBoolean("silentMode",false);
       setSilent(mSilentMode); // Do something with silent
    }

    @Override
    protectedvoid onStop(){
       super.onStop();

      // We need an Editor object to make preference
changes.
      SharedPreferences settings =
```

```
getSharedPreferences(PREFS_NAME,0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("silentMode", mSilentMode);

      // Commit the edits!
      editor.commit();
    }
}
```

. أنظر الدالة ()onStop ولاحظ كيف تم تطبيق الخطوات في الأعلى لحفظ قيمة المتغير mSilentMode حيث تم استخدام المفتاح "silentMode" كمعرف. لاحظ أن ملف التفضيلات محدد باسم MyPrefsFile والذي تم تمريره للدالة ()getSharedPreferences. عند تشغيل الفعالية مرة أخرى، يتم استرجاع البيانات من التفضيلات المشتركة أثناء تنفيذ الدالة ()onCreate. لاحظ أنه تم الرجوع إلى نفس الملف والذي اسمه MyPrefsFile.

ملاحظة: عند تشغيل الفعالية لأول مرة لن يكون هناك بيانات مخزنة خاصة بالفعالية. في هذه الحالة سيأخذ المتغيرالقيمة الافتراضية وهي false والتي تم تحديدها في المدخل الثاني للدالة getBoolean("silentMode", false).

الذاكرة الداخلية (Internal Storage)

يمكن تخزين البيانات مباشرةً في الذاكرة الداخلية للجهاز، وتكون البيانات المخزنة خاصة بالتطبيق الذي تم التخزين من خلاله ولا يمكن للتطبيقات الأخرى الوصول لهذه البيانات. عند حذف التطبيق تحذف البيانات المتعلقة به تلقائياً.

لإنشاء وتخزين ملف في الذاكرة الداخلية يمكن استعمال الخطوات التالية:

1. نفذ الدالة ()openFileOutput ومرر لها اسم الملف المراد انشاؤه وتخزينه ووضع الوصول Operating mode. تقوم هذه الدالة بإنشاء الملف وإرجاع كائن من نوع FileOutputStream والذي من خلاله يتم الكتابة إلى الملف.
2. نظراً لأن FileOutputStream يستخدم لكتابة البيانات بالصيغة الأولية Bytes، و حتى تتمكن من كتابة أي نص، أنشي كائن من نوع PrintWriter ومرر له الكائن من نوع FileOutputStream.
3. اكتب إلى الملف باستخدام الدالة ()print.
4. أغلق الكائنات من نوعFileOutputStream و PrintWriter ذلك حتى لا يبقى الملف مفتوحاً بعد الانتهاء منه.

هذه الخطوات موضحة في الكود التالي:

```
String fileName ="hello_file";
String string="hello world!";
FileOutputStream fos =
openFileOutput(fileName,Context.MODE_PRIVATE);
PrintWriter pr = new PrintWriter(fos);
pr.print(string);
pr.close();
fos.close();
```

لاحظ أن وضع الوصول MODE_PRIVATE يعني أن الوصول للملف خاص بالتطبيق الحالي فقط.

لقراءة ملف سبق تخزينه في الذاكرة الداخلية يمكن اتباع الخطوات التالية:

1. نفذ الدالة openFileInput() ومرر لها اسم الملف المراد فتحه. هذه الدالة تقوم بإرجاع كائن من نوع FileInputStream.
2. حتى تتمكن من قراءة محتويات الملف سطراً سطراً، أنشئ كائن من نوع BufferedReader ومرر له الكائن FileInputStream
3. اقرأ محتويات الملف سطراً سطراً باستخدام الدالة readLine حتى تصل إلى نهاية الملف (عندها ترجع الدالة readLine قيمة null).
4. أغلق الكائنات من نوع BufferedReader و FileInputStream.

هذه الخطوات موضحة بالكود التالي:

```
FileInputStream in = openFileInput("hello_file");
BufferedReader br = new BufferedReader(new
InputStreamReader(in));
String strLine;
while((strLine = br.readLine())!= null){
System.out.println(strLine);
}
br.close();
in.close();
```

الذاكرة الخارجية (External Storage)

يدعم نظام أندرويد حفظ الملفات في الذاكرة الخارجية والتي قد تكون ذاكرة يمكن إزالتها وتركيبها removable مثل SD Card، أو ذاكرة ثابتة لا يمكن إزالتها. الملفات المخزنة في الذاكرة الخارجية يمكن الوصول إليها من أي تطبيق آخر وذلك بعكس الملفات المخزنة بالذاكرة الداخلية والتي تكون خاصة بالتطبيق.

قبل القراءة أو الكتابة للذاكرة الخارجية يجب إضافة إذن (permission) في ملف الوثيقة (Manifest) الخاص بالتطبيق، حيث أن هناك إذن بالقراءة (READ_EXTERNAL_STORAGE) و إذن بالكتابة (WRITE_EXTERNAL_STORAGE) كما هو موضح بالأسفل:

```
<manifest ...>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  ...
</manifest>
```

ملاحظة: ابتداء من Android 4.4 لا تحتاج لإذن permission إذا أردت كتابة أو قراءة ملفات خاصة بالتطبيق فقط.

كما يجب قبل محاولة الوصول للذاكرة الخارجية التأكد من أن الذاكرة متوفرة عن طريق تنفيذ الدالة getExternalStorageState(). الذاكرة الخارجية قد تكون في حالة "مركبة" Mounted، "مفقودة" missing, "القراءة فقط" read only بالإضافة لحالات أخرى.المثال التالي يوضح دالتين لفحص حالة الذاكرة الخارجية: الأولى تفحص إذا كانت الذاكرة متوفرة للكتابة والقراءة بينما تفحص الثانية إذا كانت الذاكرة متوفرة للقراءة فقط.

```
/* Checks if external storage is available for read and
write */
public boolean isExternalStorageWritable(){
    String state =Environment.getExternalStorageState();
    if(Environment.MEDIA_MOUNTED.equals(state)){
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read
*/
public boolean isExternalStorageReadable(){
    String state =Environment.getExternalStorageState();
    if(Environment.MEDIA_MOUNTED.equals(state)||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
        return true;
    }
    return false;
}
```

المثال التالي يوضح كيفية تخزين ملف في الذاكرة الخارجية. الدالة Environment.getExternalStorageDirectory() تقوم بإرجاع مسار المجلد الذي يمثل الذاكرة الخارجية حيث يمكن حفظ الملفات.

```
File directory = Environment.getExternalStorageDirectory();
File file = new File(directory, "externalFile.txt");
if(!file.exists())
     file.createNewFile();
FileOutputStream fos = new FileOutputStream(file);
OutputStreamWriter osr = new OutputStreamWriter(fos);
PrintWriter pr = new PrintWriter(osr);
pr.write();
osr.close();
fos.close();
```

تمرين عملي (1-6)