الفصل السادس:

حالة النشاط، التفضيلات، الملفات، الكاميرا

الكلمات المفتاحية:

حالة منتسخ النشاط، الحالة غير الدائمة والحالة الدائمة لنشاط، حفظ الحالة، الملفات، الكاميرا.

ملخص:

نُبين أولاً حالات النشاط المختلفة وكيفية فقدان حالة النشاط ثم نستعرض آليات حفظ حالة النشاط غير الدائمة وحالة النشاط الدائمة. نتعرض بعد ذلك لآليات التعامل مع الملفات ثم التعامل مع الكاميرا.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- إدارة حالة النشاط.
- استخدام التفضيلات.
- التعامل مع الملفات.
- التعامل مع الكاميرا

المخطط:

حالة النشاط، التفضيلات، الملفات، الكاميرا

- 4 وحدات (Learning Objects)

1. حالة النشاط

الأهداف التعليمية:

- حالة النشاط

حالة منتسخ النشاط Activity instance state

- ندعو حالة منتسخ النشاط الحالة الحالية لنشاط. مثلاً:
  - النصوص المُدخلة في صناديق تحرير النصوص.
  - حالة صندوق تحقق.
  - الزر المُحدّد من مجموعة أزرار خيار.
  - ...

مثلاً، حالة النشاط في الواجهة التالية:

- زر الراديو Dan هو المحدّد.
- صورة Dan هي الصورة الظاهرة.

ضياع حالة النشاط Lost activity state

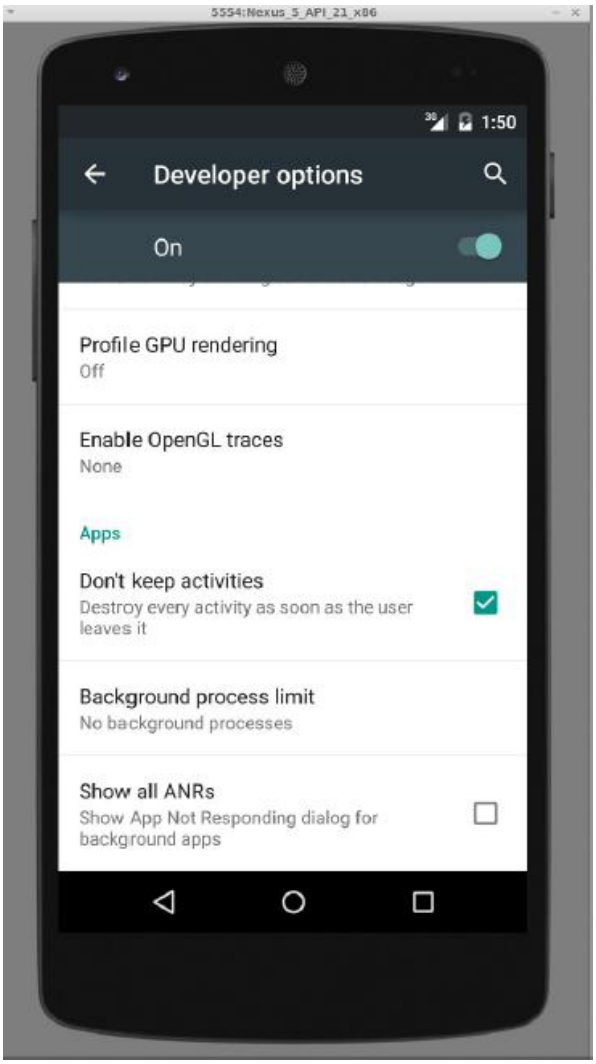
- يُمكن خسارة حالة منتسخ النشاط في أحيان كثيرة. مثلاً:
  - الخروج من النشاط إلى نشاط آخر ومن ثم العودة إليه.
  - استدعاء تطبيق آخر app ومن ثم العودة للنشاط الأول.
  - تغيير اتجاه الجهاز.

محاكاة تغير الحالة في الجهاز الافتراضي Simulating state change in AVD

- تذكر أنه يُمكنك تغيير حالة الاتجاه (طولي أم عرضي) بضغط Ctrl+F11.
- لاختبار إنهاء النشاطات (الحدث onDestroy)، قم بالدخول إلى الإعدادات:

Settings → Developer options → Don't keep activities

Developer options → Background process limit → No background processes

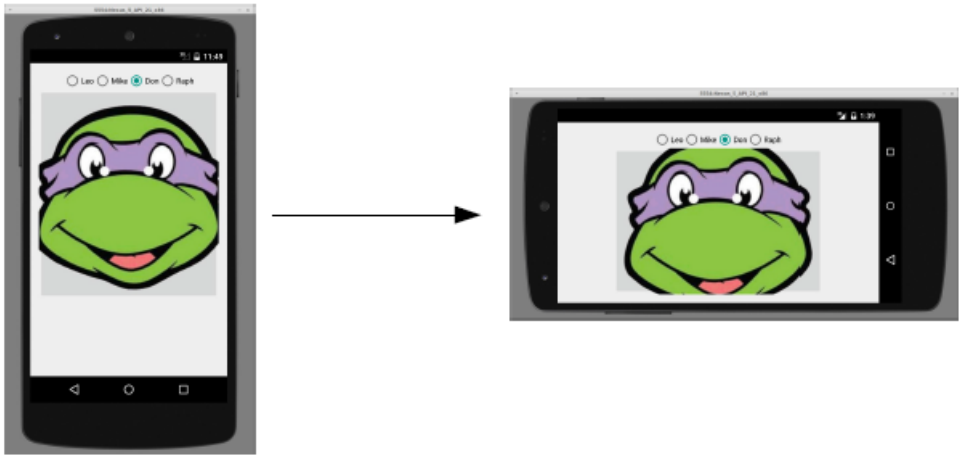


معالجة الدوران Handling rotation

- ملاحظة: يوجد طريقة سريعة للمحافظة على حالة النشاط عند تغيير اتجاه الجهاز وذلك بكتابة الإعدادات التالية في الملف AndroidManifest.xml: (لا يحل ذلك الحالات الأخرى مثل تحميل تطبيقات أو نشاطات أخرى).

```
<activity android:name=".MainActivity"

android:configChanges="orientation|screenSize"...>
```



الحالة غير الدائمة والحالة الدائمة لنشاط non-persistent/permanent state

- ندعو الحالة غير الدائمة لنشاط حالة النشاط التي لا تدوم بعد إنهاء التطبيق كقيم عناصر التحكم على الواجهات بشكل عام. أما القيم التي نريد أن نحافظ عليها بشكل دائم فندعوها بالقيم الدائمة permanent كإعدادات التطبيق بشكل عام (والتي يُمكن طبعاً أن يُغيرها المستخدم) والبيانات المدخلة في عناصر التحكم.

الحفاظ على الحالة غير الدائمة لنشاط non-persistent state

- يتم الحفاظ على الحالة غير الدائمة لنشاط باستخدام الإجرائية onSaveInstanceState واستعادة الحالة باستخدام الإجرائية onRestoreInstanceState.

- تُستدعى الإجرائية onSaveInstanceState بعد إنهاء النشاط ولها معامل من النمط Bundle والذي يسمح بتخزين أزواج مفتاح/قيمة key/value.

- يتم في هذه الإجرائية تخزين الأزواج مفتاح/قيمة المطلوبة:

```
public void onSaveInstanceState(Bundle outState) {
super.onSaveInstanceState(outState); // always call super
outState.putInt("name", value);
outState.putString("name", value);
...
}
```

- تُستدعى الإجرائية onRestoreInstanceState بعد إعادة إنشاء النشاط ولها معامل من النمط Bundle (الذي تم تخزين الأزواج فيه في الإجرائية onSaveInstanceState).

```
public void onRestoreInstanceState(Bundle inState) {
super.onRestoreInstanceState(inState); // always call super
int name = inState.getInt("name");
String name = inState.getString("name");
...
}
```

Saving your own classes حفظ صفوفك الخاصة

- لا يُمكنك بشكل افتراضي تخزين الأغراض من صفوفك المخصصة في Bundle. للقيام بذلك، يجب أن يُنفذ صفك الواجهة java.io.Serializable() واستخدام الطريقة putSerializable.

```
public class Date implements Serializable {
...
}
public class MainActivity extends Activity {
public void onSaveInstanceState(Bundle outState) {
super.onSaveInstanceState(outState);
Date d = new Date(2015, 1, 25);
outState.putSerializable("today", d);
   }
}
```

مثال عملي

- نقوم في المثال التالي بتخزين قيمة زر الخيار المُحدّد من مجموعة من أزرار الخيار:

```
protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

Log.d("testing", "onSaveInstanceState got called");

RadioGroup group =
       (RadioGroup)findViewById(R.id.turtle_group);

int id = group.getCheckedRadioButtonId();

outState.putInt("id", id);

}
```

- ثم نقوم باسترجاع القيم المخزنة:

```
protected void onRestoreInstanceState
                        (Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

Log.d("testing", "onRestoreInstanceState got called");



int id = savedInstanceState.getInt("id");

RadioGroup group =
        (RadioGroup)findViewById(R.id.turtle_group);

group.check(id);

updateTurtleImage();



}
```

الحفاظ على القيم بشكل دائم باستخدام التفضيلات Preferences

- يوفر الغرض SharedPreferences إمكانية حفظ أزواج مفتاح/قيمة بشكل دائم ولو بعد إغلاق التطبيق.
- يوجد حالتين:
  - إما حفظ القيم على مستوى النشاط (per-activity) فنستخدم getPreferences.
  - أو حفظ القيم على مستوى التطبيق (per-app) فنستخدم getPreferences مع تمرير اسم ملف لهذه الطريقة.

- لحفظ القيم على مستوى النشاط:

```
SharedPreferences prefs = getPreferences(MODE_PRIVATE);
SharedPreferences.Editor prefsEditor = prefs.edit();
prefsEditor.putInt("name", value);
prefsEditor.putString("name", value);
...
prefsEditor.apply(); // or commit();
```

- لاستعادة القيم:

```
SharedPreferences prefs = getPreferences(MODE_PRIVATE);
int i = prefs.getInt("name", defaultValue);
String s = prefs.getString("name", "defaultValue");
...
```

- لحفظ القيم على مستوى التطبيق في ملف:

```
SharedPreferences prefs = getSharedPreferences
                                ("filename", MODE_PRIVATE);
SharedPreferences.Editor prefsEditor = prefs.edit();
prefsEditor.putInt("name", value);
prefsEditor.putString("name", value);
...
prefsEditor.commit();
```

- لاستعادة القيم:

```
SharedPreferences prefs = getSharedPreferences
                                ("filename", MODE_PRIVATE);
```

```
int i = prefs.getInt("name", defaultValue);

String s = prefs.getString("name", "defaultValue");

...
```

2. التعامل مع الملفات

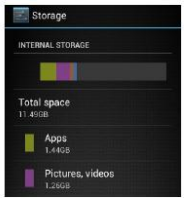
الأهداف التعليمية:

- التعامل مع الملفات

التخزين الداخلي والتخزين الخارجي Internal and external storage

- يُمكن في أندرويد قراءة/كتابة الملفات من مكاني التخزين: الداخلي والخارجي واللذان يحافظان بشكل دائم على القيم persistent storage (ولو تمّ إطفاء الجهاز power-off أو إعادة تشغيله reboot).

التخزين الداخلي Internal storage



يتميز بما يلي:

- تخزين دائم يتم على الجهاز نفسه.
- صغير نسبياً (1–4 غيغا بايت).
- لا يُمكن توسعته أو إزالته.
- مُحدّد وخاص لكل تطبيق.
- يتم مسحه عند إلغاء تنصيب التطبيق.

الملف File

- يُمثّل الملف أو المجلد بأغراض من الصف java.io.File.
- من أهم طرق هذا الصف:

canRead, canWrite, create, delete, exists, getName, getParent, getPath, isFile, isDirectory, lastModified, length, listFiles, mkdir, mkdirs, renameTo

الدفقات Streams

- تُمثّل تدفقات بايتات البيانات من/إلى المصدر أو الوجهة.
- نستخدم الصفين: java.io.InputStream و java.io.OutputStream.
- يُمكن أن تأتي من ملف، شبكة، قاعدة بيانات، الذاكرة، ...
- تحوي طرق لقراءة/كتابة بايت (محرف) من الدخل.
- تُمرر عادة كمعاملات لأغراض أخرى لإجراء القراءة/الكتابة فعلياً مثل:

java.util.Scanner, java.io.BufferedReader, java.io.PrintStream.

استخدام التخزين الداخلي Using internal storage

- للنشاط مجموعة من الطرق يُمكن استخدامها لقراءة/كتابة الملفات:

- `getFilesDir()` – returns internal directory for your app
- `getCacheDir()` – returns a "temp" directory for scrap files
- `getResources().openRawResource(R.raw.`***id***`)`
  -read an input file from `res/raw/`
- `openFileInput("`***name***`", `***mode***`)` – opens a file for reading
- `openFileOutput("`***name***`", `***mode***`)` – opens a file for writing

- يُعيد العديد من الطرق أغراض من الصف java.io.File.
- تُعيد بعض الطرق أغراض من java.io.InputStream أو OutputStream والتي يُمكن استخدامها لكتابة/قراءة الملفات مع الصفوف القياسية:

Scanner, BufferedReader, PrintStream.

أمثلة تطبيقية

- يُبين المثال التالي كيفية قراءة ملف موجود في المصادر وعرض محتواه:

```
// read a file, and put its contents into a TextView
// (assumes hello.txt file exists in res/raw/ directory)
Scanner scan = new Scanner
        (getResources().openRawResource(R.raw.hello));
String allText = ""; // read entire file
while (scan.hasNextLine()) {
 String line = scan.nextLine();
 allText += line;
}
myTextView.setText(allText);
scan.close();
```

- مثال كتابة/قراءة:

```
// write a short text file to the internal storage
PrintStream output = new PrintStream
          (openFileOutput("out.txt", MODE_PRIVATE));
output.println("Hello, world!");
output.println("How are you?");
output.close();
 ...
// read the same file, and put its contents into a TextView
Scanner scan = new Scanner
  (openFileInput("out.txt", MODE_PRIVATE));
String allText = ""; // read entire file
while (scan.hasNextLine()) {
        String line = scan.nextLine();
        allText += line;
```

```
}

myTextView.setText(allText);

scan.close();
```

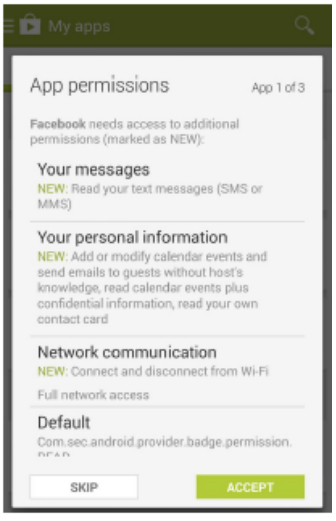
التخزين الخارجي External Storage

وهي بطاقات التخزين التي توصل مع الجهاز وتتميز بما يلي:

- ذات سعات كبيرة (4–32 غيغا بايت).
- يُمكن نقلها من جهاز لجهاز آخر.
- غير خاصة بالتطبيق إذ يُمكن القراءة/الكتابة من تطبيقات أخرى ومستخدمين آخرين.
- يُمكن ألا تكون موجودة على الجهاز.
- لا يتم مسحها بشكل عام عند إلغاء تنصيب التطبيق.
- يجب إعطاء التطبيق السماحية باستخدامها في الملف **AndroidManifest.xml** بكتابة:

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

- عند تنزيل التطبيق، سيتم الطلب من المستخدم التأكيد على سماحية الاستخدام.



- من أهم الطرق للقراءة/الكتابة على التخزين الخارجي:
- الطريقة:

getExternalFilesDir("***name***")

والتي تُعيد مجلد خارجي خاص (private) لتطبيقك بالاسم المحدد.

- الطريقة:

Environment.getExternalStoragePublicDirectory(***name***)

والتي تُعيد مجلد عام (public) للملفات العامة مثل الصور والموسيقا.

يُمرر للطريقة ثابت من واحد من الثوابت التالية:

Environment.DIRECTORY_ALARMS,
DIRECTORY_DCIM, DIRECTORY_DOWNLOADS, DIRECTORY_MOVIES,
DIRECTORY_MUSIC, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES,
DIRECTORY_PODCASTS, DIRECTORY_RINGTONES

- يُبين المثال التالي مثلاً كيفية الوصول إلى مجلد الصور ومن ثم الدوران عليها:

```
// write short data to app-specific external storage
File outDir = getExternalFilesDir(null); // root dir
File outFile = new File(outDir, "example.txt");
PrintStream output = new PrintStream(outFile);
output.println("Hello, world!");
output.close();
// read list of pictures in external storage
        File picsDir =
        Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES);
        for (File file : picsDir.listFiles()) {
        ...
        }
```

التعامل مع بيانات الوب Accessing web data

- لقراءة البيانات من الوب يجب إعطاء طلب سماحية INTERNET في الملف AndroidManifest.xml:

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

- يُبين المثال التالي كيفية قراءة بيانات ملف من الوب:

```
URL url = new URL("http://foobar.com/example.txt");
Scanner scan = new Scanner(url.openStream());
while (scan.hasNextLine()) {
String line = scan.nextLine();
...
}
```

2. التعامل مع الكاميرا

الأهداف التعليمية:

- التعامل مع الكاميرا.

طلب سماحية استخدام الكاميرا

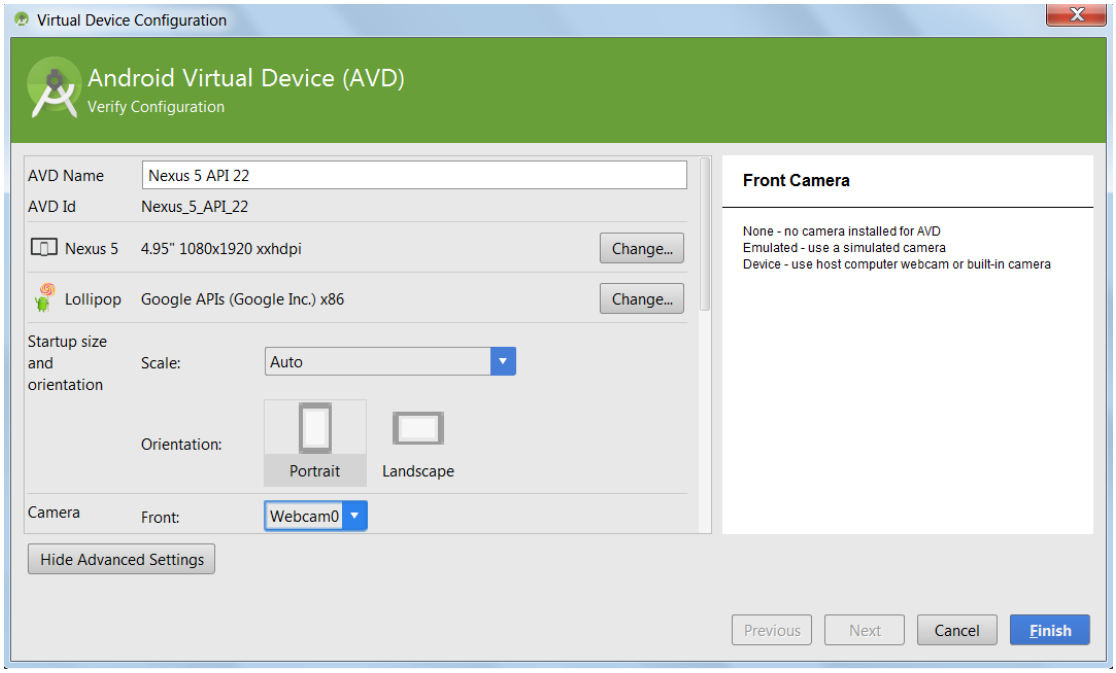
- لاستخدام الكاميرا في التطبيق، يجب طلب السماحية في الملف AndroidManifest.xml:

```
<manifest ...>
<uses-permission
android:name="android.permission.CAMERA" />
<uses-feature
android:name="android.hardware.camera"
android:required="true" />
...
</manifest>
```

الكاميرا في المحاكي

- لاستخدام الكاميرا في المحاكي يجب ضبط الإعدادات اللازمة:

AVD Manager → Edit → Advanced Settings → Camera



مثال تطبيقي

- يُبين المثال التالي كيفية التقاط صورة من الكاميرا ومن ثم وضعها في عنصر عرض صورة ImageView:

```
public class MainActivity extends Activity {

  // these "request codes" are used to identify sub-activities that return results

  private static final int REQUEST_CODE_DETAILS_ACTIVITY = 1234;

  private static final int REQUEST_CODE_TAKE_PHOTO = 4321;



  private static final int REQ_CODE_TAKE_PICTURE = 90210;

  ...

  Intent picIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

  startActivityForResult(picIntent, REQ_CODE_TAKE_PICTURE);

  ...

 @Override

protected void onActivityResult
                    (int requestCode, int resultCode,  Intent intent) {

    if (requestCode == REQ_CODE_TAKE_PICTURE

          && resultCode == RESULT_OK) {

        Bitmap bmp = (Bitmap) intent.getExtras().get("data");

        ImageView img = (ImageView) findViewById(R.id.camera_image);

        img.setImageBitmap(bmp);

    }

  }
```

صف الكاميرا

- للحصول على تحكم أكبر بالكاميرا، يُمكن استخدام صف الكاميرا والذي يوفر مجموعة من الطرق مثل:

```
Camera cam = Camera.open();
Camera.Parameters params = cam.getParameters();
params.setPreviewSize(w, h);
cam.setParameters(params);
cam.setDisplayOrientation(degrees); // degrees
cam.startPreview();
cam.takePicture(...); // pass "callbacks" to run later
cam.stopPreview();
cam.release();
```

3. مثال تعليمي

الأهداف التعليمية:

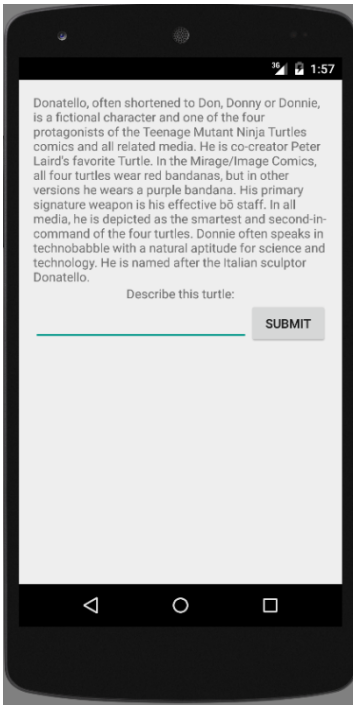
- مثال تعليمي

مثال تعليمي

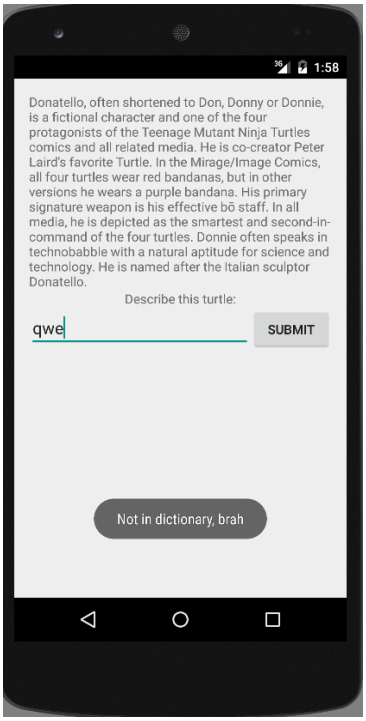
- عند تشغيل التطبيق تظهر الواجهة التالية:

- عند النقر على أحد أزرار الراديو تظهر صورة الشخصية الموافقة:

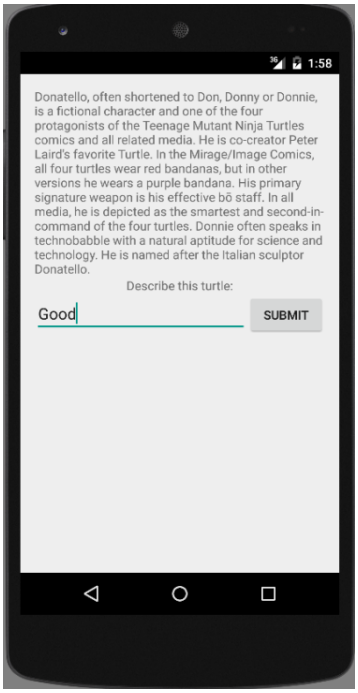
- عند النقر على صورة شخصية، ننتقل إلى الواجهة الثانية حيث يتم عرض بيانات الشخصية الموافقة والطلب من المستخدم كتابة تقويم للشخصية:



- في حال قام المستخدم بكتابة كلمة غير موجودة في "قاموس" التطبيق تظهر رسالة خطأ موافقة:



- أما إذا قام المستخدم بكتابة كلمة موجودة في القاموس:

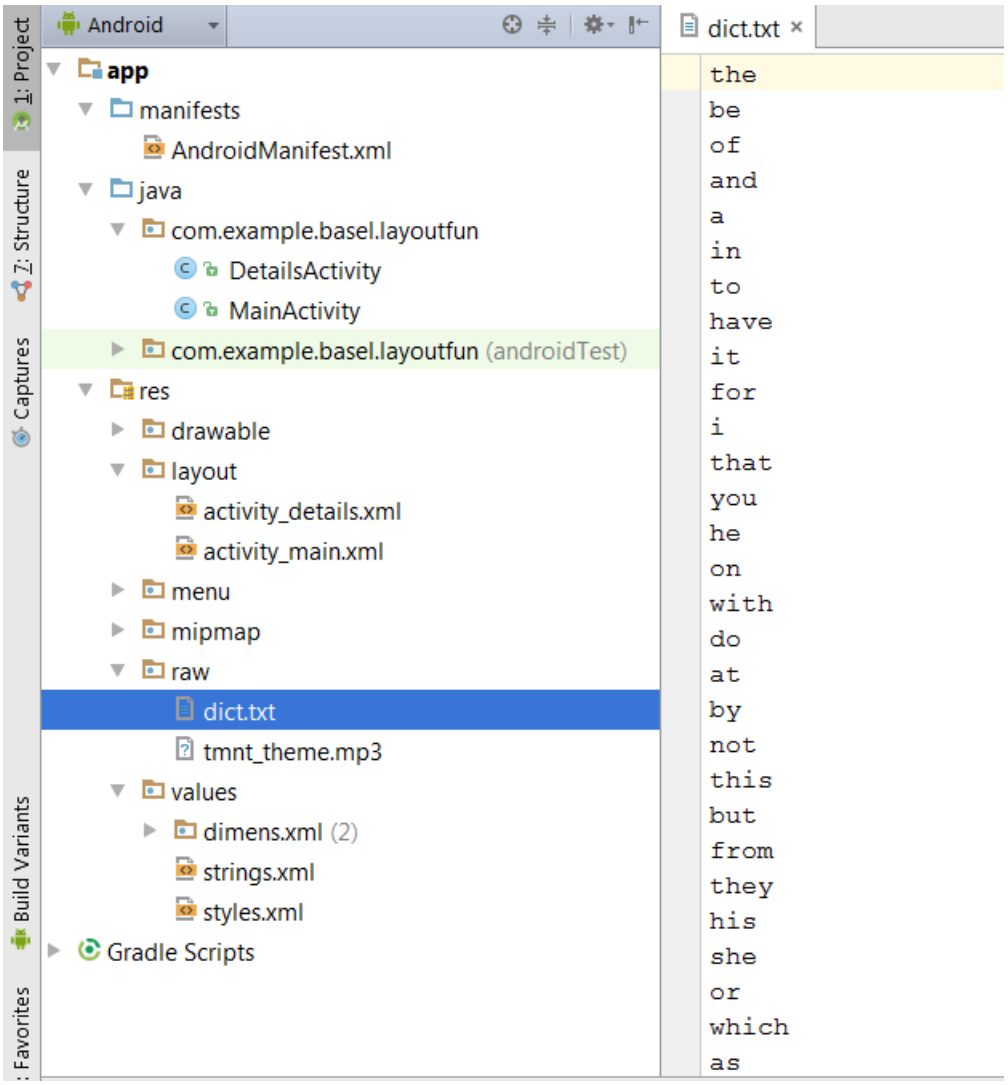


- يتم العودة إلى الواجهة الأولى مع إظهار رسالة التقويم:

- كما يُمكن التقاط صورة عن طريق زر الأمر Take Photo.

تصميم التطبيق

- يحوي الملف (القاموس) raw\dict.txt مجموعة من كلمات اللغة الإنكليزية:



- يكون تصميم النشاط الأول activity_main.xml:

```
<LinearLayout
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:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:gravity="top|center"

    android:orientation="vertical"

    android:paddingRight="@dimen/activity_horizontal_margin"

    android:paddingTop="@dimen/activity_vertical_margin"

    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">



<RadioGroup

        android:id="@+id/turtle_group"

        android:orientation="horizontal"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content">

<RadioButton

            android:id="@+id/leo"

            android:onClick="pickTurtle"

            android:text="Leo"

            android:checked="true"

            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content" />



<RadioButton

            android:id="@+id/mike"

            android:onClick="pickTurtle"

            android:text="Mike"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content" />



<RadioButton

            android:id="@+id/don"

            android:onClick="pickTurtle"

            android:text="Don"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content" />



<RadioButton
```

```
            android:id="@+id/raph"

            android:onClick="pickTurtle"

            android:text="Raph"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content" />

</RadioGroup>



<ImageButton

        android:id="@+id/turtle"

        android:onClick="onClickTurtleImage"

        android:src="@drawable/tmntleo"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content" />



<Button

        android:text="Take Photo"

        android:onClick="onClickTakePhoto"
```

```
        android:layout_width="wrap_content"

        android:layout_height="wrap_content" />

</LinearLayout>
```

- ويكون كود النشاط الأول MainActivity.java:

```
package com.example.basel.layoutfun;

/*

 * onSaveInstanceState to make sure that the turtle is

 * not forgotten when the user leaves and returns to the activity.

 */



import android.app.Activity;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Bitmap;

import android.media.MediaPlayer;

import android.os.Bundle;

import android.provider.MediaStore;

import android.util.Log;

import android.view.View;
```

```
import android.widget.ImageButton;

import android.widget.RadioGroup;

import android.widget.Toast;



public class MainActivity extends Activity {

    // these "request codes" are used to identify sub-activities that return results

private static final int REQUEST_CODE_DETAILS_ACTIVITY = 1234;

private static final int REQUEST_CODE_TAKE_PHOTO = 4321;



private MediaPlayer player;    // media player for playing TMNT music



/*

* Called when the activity first gets created.

*/

@Override

protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

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

```
        player = MediaPlayer.create(this, R.raw.tmnt_theme);

        Log.d("testing", "onCreate got called; Bundle=" +
savedInstanceState);

    }



public void onPause() {

        super.onPause();

        player.stop();

        Log.d("testing", "onPause got called");

    }



public void onResume() {

        super.onResume();

        if (player != null) {

            player.setLooping(true);

            player.start();

        }

        Log.d("testing", "onResume got called");
```

```
    }



public void onStart() {

        super.onStart();

        Log.d("testing", "onStart got called");

    }



public void onStop() {

        super.onStop();

        Log.d("testing", "onStop got called");

    }



public void onDestroy() {

        super.onDestroy();

        Log.d("testing", "onDestroy got called");

    }
```

```
/*
* Called when the activity is stopped and wants to save its state.
* We save which turtle is selected and showing so that the app stays on that turtle
* when the user comes back later.
*/
@Override
protected void onSaveInstanceState(Bundle outState) {
super.onSaveInstanceState(outState);
Log.d("testing", "onSaveInstanceState got called");
RadioGroup group= (RadioGroup)findViewById(R.id.turtle_group);
int id = group.getCheckedRadioButtonId();
outState.putInt("id", id);
}


/*
* Called when the activity returns to an active running state and wants to restore its state.
* We restore which turtle was previously selected.
*/
@Override
```

```
protected void onRestoreInstanceState
         (Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

Log.d("testing", "onRestoreInstanceState got called");

int id = savedInstanceState.getInt("id");

RadioGroup group (RadioGroup)findViewById(R.id.turtle_group);

group.check(id);

updateTurtleImage();

    }



/*

* Called when the user clicks on the large TMNT image button.

* Loads the DetailsActivity for more information about that turtle.

*/

public void onClickTurtleImage(View view) {

Intent intent = new Intent(this, DetailsActivity.class);

RadioGroup group=(RadioGroup) findViewById(R.id.turtle_group);

 int id = group.getCheckedRadioButtonId();
```

```
 intent.putExtra("turtle_id", id);

 startActivityForResult(intent,REQUEST_CODE_DETAILS_ACTIVITY);

}



/*

* Event handler that is called when a sub-activity returns.

* We can extract the data that came back from the other activity

* (packed into the Intent that it sends back) and use it here.

*/

@Override

protected void onActivityResult(int requestCode, int resultCode,
Intent intent) {

super.onActivityResult(requestCode, resultCode, intent);

if (requestCode == REQUEST_CODE_DETAILS_ACTIVITY &&

                resultCode == RESULT_OK) {

// returned from DetailsActivity; user sent us a dictionary word, so
show it as a Toast

String word = intent.getStringExtra("the_word");

Toast.makeText(this, "You typed: " + word,
Toast.LENGTH_SHORT).show();
```

```
        }
else if (requestCode == REQUEST_CODE_TAKE_PHOTO

                && resultCode == RESULT_OK) {

// returned from taking a photo with the camera; grab the photo and
show it

Bitmap bmp = (Bitmap) intent.getExtras().get("data");

ImageButton img = (ImageButton)findViewById(R.id.turtle);

img.setImageBitmap(bmp);

        }

}



/*

* This method is called when the user chooses one of the turtle radio buttons.

* In this code we set which turtle image is visible on the screen in the ImageView.

*/

public void pickTurtle(View view) {

        updateTurtleImage();

    }
```

```
/*
* Called when the "Take Photo" button is clicked.
* Launches a new activity to take a photo using the camera.
*/
public void onClickTakePhoto(View view) {
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, REQUEST_CODE_TAKE_PHOTO);
}


/*
* Called by various event handlers to update which turtle image is showing
* based on which radio button is currently checked.
*/
private void updateTurtleImage() {
ImageButton img = (ImageButton) findViewById(R.id.turtle);
RadioGroup group=(RadioGroup)findViewById(R.id.turtle_group);
        int checkedID = group.getCheckedRadioButtonId();
        if (checkedID == R.id.leo) {
```

```
            img.setImageResource(R.drawable.tmntleo);

        } else if (checkedID == R.id.mike) {

            img.setImageResource(R.drawable.tmntmike);

        } else if (checkedID == R.id.don) {

            img.setImageResource(R.drawable.tmntdon);

        } else if (checkedID == R.id.raph) {

            img.setImageResource(R.drawable.tmntraph);

        }

    }

}
```

- يكون تصميم النشاط الثاني activity_details.xml:

```
<LinearLayout
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:paddingLeft="@dimen/activity_horizontal_margin"

    android:gravity="top|center"

    android:orientation="vertical"

    android:paddingRight="@dimen/activity_horizontal_margin"

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

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.basel.layoutfun.DetailsActivity">

<TextView
        android:id="@+id/turtle_info"
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

<TextView
        android:text="Describe this turtle:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/the_word"
            android:layout_weight="1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

<Button
```

```
            android:text="Submit"
            android:onClick="onclickSubmit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
</LinearLayout>
</LinearLayout>
```

- ويكون كود النشاط الثاني DetailsActivity.java:

```
package com.example.basel.layoutfun;
/*
 * This file represents the Java code for the second activity.
 * This version adds a dictionary for looking up words.
 */

import android.app.Activity;
import android.content.*;
import android.os.*;
import android.support.v7.app.*;
import android.view.*;
import android.widget.*;
import java.util.*;

public class DetailsActivity extends Activity {
private Set<String> dictionary;// dictionary of words to look up
```

```
/*
* Constant array of data about each of the four turtles.
* (This is not the most idiomatic way to store such information,
*/
private static final String[] TURTLE_DETAILS = {

            "Leonardo, or Leo, is one of the four protagonists of the Teenage
Mutant Ninja Turtles comics and all related media. In the Mirage/Image Comics, all
four turtles wear red bandanas, but in other versions, he wears a blue bandana.
His signature weapons are two ninjato. Throughout the various media, he is often
depicted as the eldest and leader of the four turtles, as well as the most
disciplined. He is named after Leonardo da Vinci. In the 2012 series, he is the
only turtle who harbors strong romantic affections for Karai, considering her his
love interest.",

            "Michelangelo, Mike or Mikey (as he is usually called), is a fictional
character and one of the four protagonists of the Teenage Mutant Ninja Turtles
comics and all related media. His mask is typically portrayed as orange outside of
the Mirage/Image Comics and his weapons are dual nunchucks, though he has also
been portrayed using other weapons, such as a grappling hook, manriki-gusari,
tonfa, and a three section staff (in some action figures).",

            "Donatello, often shortened to Don, Donny or Donnie, is a fictional
character and one of the four protagonists of the Teenage Mutant Ninja Turtles
comics and all related media. He is co-creator Peter Laird's favorite Turtle. In
the Mirage/Image Comics, all four turtles wear red bandanas, but in other versions
he wears a purple bandana. His primary signature weapon is his effective bō staff.
In all media, he is depicted as the smartest and second-in-command of the four
turtles. Donnie often speaks in technobabble with a natural aptitude for science
and technology. He is named after the Italian sculptor Donatello.",

            "Raphael, or Raph, is a fictional character and one of the four
protagonists of the Teenage Mutant Ninja Turtles comics and all related media. In
the Mirage/Image Comics, all four turtles wear red bandanas over their eyes, but
unlike his brothers in other versions, he is the only one who keeps the red
bandana. Raphael wields twin sai as his primary weapon. (In the Next Mutation
series, his sai stick together to make a staff-like weapon.) He is generally the
most likely to experience extremes of emotion, and is usually depicted as being
aggressive, sullen, maddened, and rebellious. The origin of Raphael's anger is not
always fully explored, but in some incarnations appears to stem partly from the
realization that they are the only creatures of their kind and ultimately alone.
He also has a somewhat turbulent relationship with his older brother Leonardo
because Leonardo is seen as the group's leader. Raphael is named after the 16th-
century Italian painter Raphael. In 2011 Raphael placed 23rd on IGN's Top 100
Comic Book Heroes, a list that did not feature any of his brothers."

    };
```

```
/*
* Called when the activity first gets created.
* Shows detail text about the selected ninja turtle.
*/
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);


// pull the turtle's ID out of the intent that the MainActivity used
to load me
        Intent intent = getIntent();
        int id = intent.getIntExtra("turtle_id", R.id.leo);
        String text = "";
        if (id == R.id.leo) {
            text = TURTLE_DETAILS[0];
        } else if (id == R.id.mike) {
            text = TURTLE_DETAILS[1];
        } else if (id == R.id.don) {
            text = TURTLE_DETAILS[2];
        } else { // if (id == R.id.raph)
            text = TURTLE_DETAILS[3];
        }
    TextView tv = (TextView) findViewById(R.id.turtle_info);
        tv.setText(text);
```

```
        if (dictionary == null) {
            loadDictionary();
        }
    }

/*
* Called when the user presses the Submit button.
* Checks whether the user has typed a legal dictionary word in the text box,
* and if so, closes this activity and returns to the MainActivity.
*/
public void onclickSubmit(View view) {
// if user has typed a valid dictionary word, accept it and send it
back to main activity
 EditText edit = (EditText) findViewById(R.id.the_word);
 String text = edit.getText().toString().trim().toLowerCase();
        if (dictionary.contains(text)) {
            Intent intent = new Intent();
            intent.putExtra("the_word", text);
            setResult(RESULT_OK, intent);
            finish();
        } else {
            Toast.makeText(this, "Not in dictionary, brah",
Toast.LENGTH_SHORT).show();
        }
    }
```

```
/*
* Loads the dictionary words from the provided text file, dict.txt.
*/
private void loadDictionary() {
dictionary = new HashSet<String>();
Scanner scan = new
Scanner(getResources().openRawResource(R.raw.dict));
        while (scan.hasNextLine()) {
            String word = scan.nextLine();
            dictionary.add(word);
        }
    }

// AUTO-GENERATED CODE

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

@Override
public boolean onOptionsItemSelected(MenuItem item) {
       // Handle action bar item clicks here. The action bar will
```

```
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();


        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }


        return super.onOptionsItemSelected(item);
    }
}
```