Programozási technológia

Java technológiai kitekintés (Android)

Dr. Szendrei Rudolf
ELTE Informatikai Kar
2020.

Android platform

- Korábban csak mobiltelefonokra, ma már laptopokra, televíziókra, Set-Top-Box-okra stb. is telepített linux alapú operációs rendszer
- Az eszközök lényegesen kevesebb erőforrással rendelkeznek, mint az asztali számítógépek
  - Alacsonyabb számítási kapacitás
  - Jelentősen kevesebb memória
  - Kisebb háttértár kapacitás
  - Tenyérnyi képernyő
  - Korlátozott üzemidő
  - Eltérő, főként virtuális adatbeviteli módok

Android platform

- Az korlátozott erőforrások miatt egyszerre csak egy alkalmazás látható az eszköz képernyőjén (kivéve…)
- A beágyazott rendszerekre jellemzően a multitasking jelentős módon korlátozott, azaz a képernyőn látható alkalmazáson kívüli programok csak nagyon ritkán, vagy egyáltalán nem kapnak processzor időt
- A fentiek alapján megkülönböztetjük a futó és futáskész alkalmazásokat
  - Futó: az épp látható, használatban lévő alkalmazás
  - Futáskész: a már elindított, de nem futó alkalmazás
- A futáskész alkalmazásokat az operációs rendszer bármikor bezárhatja, ha erőforrásra van szüksége

Android platform

- Ha a képernyőt lezárjuk, akkor az éppen futó alkalmazás már nem lesz látható → futáskész lesz
- Kivételek
  - A rendszerszintű alkalmazások mindig előnyt élveznek (pl.: telefonálás funkció)
  - A szolgáltatások a háttérben folyamatosan futhatnak (pl.: cloud szolgáltatások: google, facebook stb.)
  - A szolgáltatások felébreszthetik a készüléket az alvó módból

Android platform – Jogosultságok

- Egy eszközön sok olyan információ található, melyet nem szeretnénk minden programmal megosztani
- Az Android operációs rendszer az információ forrásokat csak megfelelő jogosultság mellett bocsájtja az adott alkalmazás rendelkezésére
- A jogosultságokat az alkalmazás a telepítése során szerezheti csak meg, a felhasználó beleegyezésével
  - Sajnos egy program jogosultságairól egyenként nem tud dönteni a felhasználó, pl.: facebook használhassa a cloud messaging-et, de a helymeghatározást és a kontaktlistát ne!

Android platform – Programok

- Ahogyan a linux platformon, úgy itt is két nagy csoportba sorolhatjuk a programokat
  - Szolgáltatások
  - Felhasználói alkalmazások (GUI-val rendelkező)
- A szolgáltatásoknak nincs grafikus felhasználói interfészük, viszont a háttérben is képesek futni
  - Pl.: üzeneteket fogad a készülék alvó módjában, vagy más GUI alkalmazások futása idején is
  - Az implementációjuk rövid, gyorsan lefutó, reaktív/reszponzív kódból kell álljon
- A felhasználói alkalmazásoknak van grafikus interfészük, de a háttérben nem futnak

Android – Programfejlesztési módok

- Az Android két fejlesztési szintet, és ezzel együtt két fejlesztési környezetet/nyelvet különböztet meg

- Alacsony szintű fejlesztés – Android NDK
  - Hatékony, CPU/GPU közeli C++ kódok készíthetők vele, tipikusan nagy számításigényű feladatok (pl.: AI, VR, Audio/Video Codec-ek stb.) gyors megoldására.

- Magas szintű fejlesztés – Android SDK ←
  - Javaban, illetve most már Kotlin nyelven is
  - Főleg gyors szoftverfejlesztésre alkalmas
  - Ebben készülnek a grafikus felületű alkalmazások, illetve a szolgáltatások túlnyomó része

Android SDK

- Célunk GUI alkalmazások készítése Java nyelven, ezért az Android SDK-t választjuk
- Az Android SDK hivatalos fejlesztői környezete az IntelliJ IDEA fejlesztő eszközre épülő Android Studio, amely több operációs rendszerre is elérhető (Windows, Linux, Mac)
- A fejlesztéshez három dolgot kell telepítenünk
  - Programozáshoz: Android Studio
  - Fordításhoz: Megfelelő Android SDK verziók a támogatandó Android verzió fényében
  - Futtatáshoz: Android emulátor vagy fizikai eszköz esetében ADB driver

Android Studio – Telepítés

- Töltsük le és telepítsük fel az Android Studio-t https://developer.android.com/studio/
- Indítsuk el, és válasszuk az egyéni beállítást
- A kinézet kiválasztása után pipáljuk ki a következőket
  - HAXM (csak Intel CPU esetén)
  - AVD (emulátor környezet)
- Sikeres település után futtassuk az SDK manager-t
  - Töröljük le a 28-as SDK-t, helyette kérjük a 22-es és 27-est
  - Emulátor telepítéséhez kérjük a 22-es és 27-es Intel Atom x86-os virtuális eszközök telepítését is (Csak HAXM esetén lesz hardveres gyorsítás velük!)

Android Studio – Teszt projekt

- A telepítés ellenőrzéséhez készítsünk egy új projektet:
  - Application name: FirstApp
  - Company domain: mycompany.hu
  - Project location: bárhova menthetjük, kivéve főkönyvtárba, és ékezeteket tartalmazó nevű könyvtárba
  - Next
  - Pipáljuk ki a Phone and Tablet opciót, és válasszuk mondjuk az API 21: Android 5.0 (Lollipop) -ot, majd Next
  - Válasszuk a Basic Activity–t
  - Next
  - Finish

Android Studio – Fordítás, Futtatás

- Ha helyesen telepítettünk mindent, akkor a fordítás a projekt létrehozása után automatikusan megtörténik.
- A projekt futtatásához kattintsunk a ▶ gombra.
- A megjelenő listában választhatunk, hogy mely csatlakoztatott eszközt, vagy mely virtuális eszközt akarjuk használni. Ha látható a listában a kívánt eszköz, akkor csak válasszuk ki és OK.
- Ha a listában nem látjuk az eszközünket, akkor azt előbb be kell konfigurálnunk.

Android Studio – Emulátor

- Emulátor beállításához nyomjuk meg a Create New Virtual Device gombot
- Kattintsunk a New Hardware Profile gombra, és adjuk meg a következő beállításokat
  - Device name: bármi lehet
  - Device type: Phone/tablet
  - Screen: 4,5" 540x960
  - RAM: 1024MB
  - Input: üresen hagyjuk
  - Finish, majd Next

Android Studio – Emulátor

- Válasszuk ki a Recommended fülön az aktuális verziót, majd Next

  (Magasabb API verzió több erőforrást igényel.
  Ha elfelejtettük volna telepíteni, a Download gombra kattintva megtehetjük most is.)

- Adjunk nevet a virtuális eszköznek.
- Állítsuk be a következőt
  - Emulated Performance → Graphics → Hardware - GLES 2.0
- Finish

Android Studio – Emulátor

- A megjelenő virtuális eszközre duplán kattintva elindul az emulátor, majd a betöltése után települ rá a programunk és el is indul.
- Az emulátor kezelőszervein a Power gomb hatására a készülék néha úgy kapcsol alvó módba, hogy nem tudjuk onnan felébreszteni újraindítás után sem. Ilyenkor tegyük a következőt
  - Android Studio → Tools → AVD Manager
  - Az eszköz sorában az Actions oszlopban a ▼ gombot használva az előugró menüből kérjük a Cold Boot Now-t
  - Az eszköz által mentett adatokat ugyanitt lehet törölni a Wipe Data segítségével

Android Studio – Fizikai eszköz

- Fizikai eszköz beállításához először engedélyeznünk kell az telefonon/tableten az USB hibakeresést
  - Beállítások → Eszköz névjegye
  - A Build számot érintsük meg 7x
- Visszalépve a beállításokhoz megjelenik a Fejlesztői beállítások menüpont. A későbbiekben itt lehet majd kikapcsolni az USB hibakeresést.
- Telepítsük az eszköz meghajtóprogramját a számítógépre. Ha nem találunk ilyet, akkor választhatunk egy általánosat is (pl.: ClockworkMod).
- Csatlakoztassuk USB-n az eszközt

Android Studio – Fizikai eszköz

- Az Android Studio újraindítása után már ki is választhatjuk a fizikai eszközt a program futtatására.
- Ahhoz, hogy a Studio az eszközön futtatni tudja a programot, telepítve kell legyen az eszköz Android verziójának megfelelő Build Tool. Amennyiben ez nincs telepítve, csak el kell fogadnunk annak az automatikus telepítését.

Android projekt

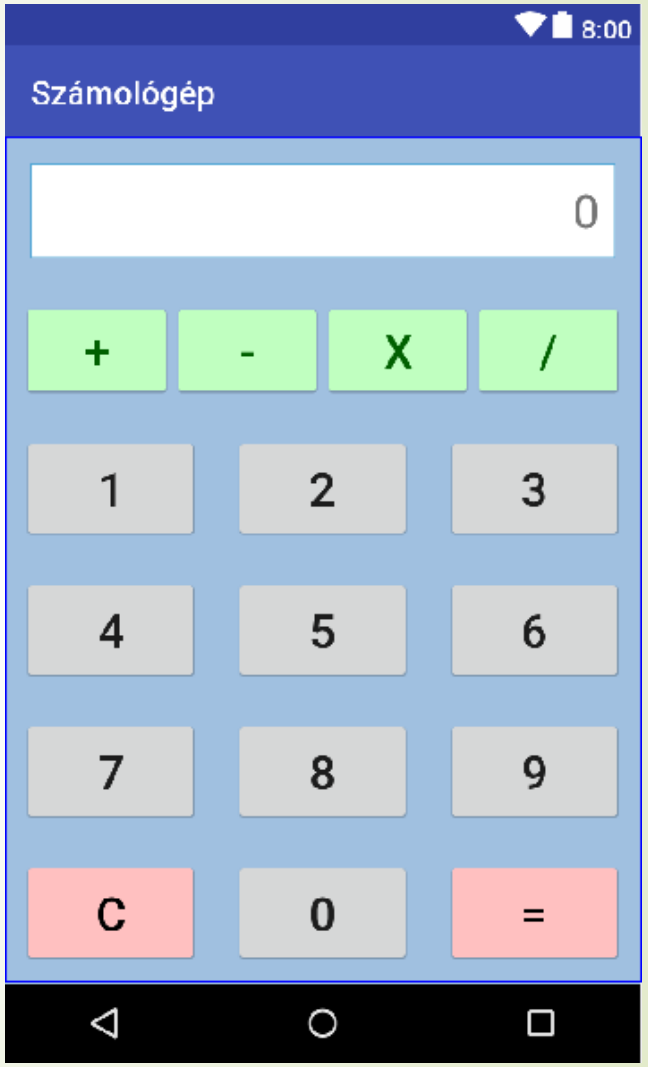
- A GUI alkalmazások az MVC architektúrát követik
  - A kinézetet XML-ben írjuk le
  - Az üzleti logikát a modell réteg tartalmazza
  - A modellt és a kinézetet a kontroller réteg kapcsolja össze
- A projekt könyvtárszerkezete
  - manifests: alkalmazás meta-adatok, jogosultságok kérése
  - java: program kódok
  - res: erőforrások
    - drawable: képek
    - layout: GUI kinézetek
    - values: színek, stílusok, menük, szövegszótárak
  - build.gradle: lib függőségek, API verzió beállítások

Számológép

- Készítsünk egy egyszerű négy alapműveletes számológépet
- Az elrendezés elkészítéséhez nyissuk meg a projektben a res → layout → activity_main.xml fájlt
- Készítsük el a számológép felületét
  - Legyen felül egy kijelzője
  - Tegyük alá a műveleteket: + - * /
  - Helyezzük el alul 3x4 helyen a számokat: 0-9
  - A számok között bal alul legyen a Clear, jobb alul az = gomb

Számológép – activity_main.xml

- A felület elemeit egy táblázatba rendezzük, ahol a kijelző és a gombok sorokban lesznek majd.
- A kijelző TextView lesz
- A gombokat Button vezérlővel valósítjuk meg



A GUI elemek paraméterei

- Mind az android: előtaggal kezdődik…
- layout_width(height) az elem dimenziója, ami lehet:
  - match_parent a befoglaló elem által adott hely
  - wrap_content a saját tartalomhoz igazodó méret
  - #dp # relatív képpontnyi méret
- layout_weight méretarány a befoglaló többi elemhez képest
- layout_margin elem és környezete közötti távolság (dp)
- padding elem és tartalma közötti távolság (dp)
- id egyedi azonosító string
- onClick eseménykezelő metódus neve
- theme alkalmazandó téma
- style alkalmazandó stílus (android: előtag nélkül!)

Számológép – stílusfájl használata

- Az elemtulajdonságok ismétlése helyett témát/stílust használunk: res\values\styles.xml

```
<style name="AppTheme.OpButton" parent="Widget.AppCompat.Button.Colored">
  <item name="android:colorButtonNormal">#c0ffc0</item>
  <item name="android:textColor">#006000</item>
</style>

<style name="AppTheme.OpButton2" parent="Widget.AppCompat.Button.Colored">
  <item name="android:colorButtonNormal">#ffc0c0</item>
  <item name="android:textColor">#000000</item>
</style>

<style name="OpButton">
  <item name="android:layout_width">0dp</item>
  <item name="android:layout_height">match_parent</item>
  <item name="android:layout_weight">1</item>
  <item name="android:layout_gravity">center_vertical</item>
  <item name="android:textSize">28dp</item>
</style>

<style name="NumButton" parent="OpButton">
  <item name="android:layout_margin">10dp</item>
  <item name="android:layout_width">0dp</item>
</style>
```

Számológép – MainActivity.java

- Itt implementáljuk a felületet példányosító kódot…

```
public class MainActivity extends AppCompatActivity {
  private TextView   txtResult;
  private Calculator calc;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtResult = findViewById(R.id.txtResult);
    calc      = new Calculator();
  }
  ...
}
```

Számológép – MainActivity.java

- …és az xml-ben már hivatkozott clickOperation-t.

```
public void clickOperation(View view) {
  char c = ((Button)view).getText().charAt(0);

  if ('0' <= c && c <= '9'){
    calc.addDigit(c - '0');
  } else {
    switch (c){
      case '=': calc.doEqual(); break;
      case 'C': calc.clear(); break;
      default:
        Operator o = Operator.getOperator(c);
        if (o != null) calc.setOperator(o); break;
    }
  }
  txtResult.setText(calc.getDisplayedText());
}
```

Számológép – Operator.java

```
public enum Operator {
  ADD('+'), SUBTRACT('-'), MULTIPLY('*'), DIVIDE('/');

  private final char op;

  Operator(char op){ this.op = op; }

  public static Operator getOperator(char c){
    switch (c) {
      case '+': return ADD;
      case '-': return SUBTRACT;
      case 'x': return MULTIPLY;
      case '/': return DIVIDE;
    }
    return null;
  }


  public char getOpChar(){ return op; }


  @Override
  public String toString(){ return op + ""; }
}
```

Számológép – Calculator.java

- A model csomagban valósítjuk meg. A számoláshoz bal és jobb operandusra, valamint egy operátorra lesz szükségünk.

```
public class Calculator {
  private int leftOperand, rightOperand;
  private Operator operator;

  public Calculator(){ clear(); }

  public void clear(){
    operator     = null;
    leftOperand  = 0;
    rightOperand = 0;
  }
  ...
}
```

Számológép – Calculator.java

```
public class Calculator {
  ...
  public void addDigit(int digit){
    rightOperand = rightOperand * 10 + digit;
  }

  public void setOperator(Operator op){
    if (operator == null){
      leftOperand  = rightOperand;
      rightOperand = 0;
    } else {
      calculate();
    }
    operator = op;
  }

  public void doEqual(){
    if (operator != null) {
      calculate();
      rightOperand = leftOperand;
      operator     = null;
    }
  }
```

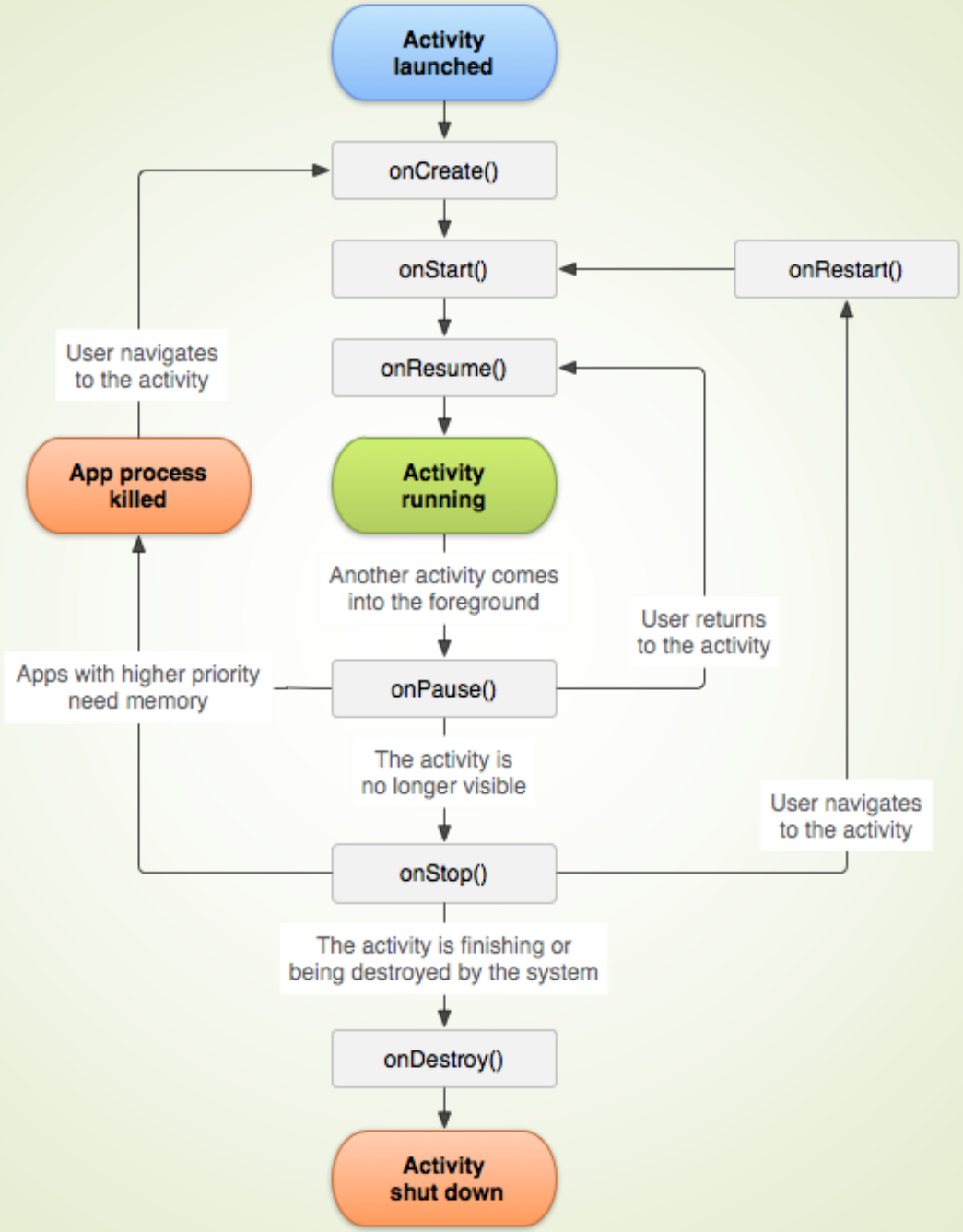
Számológép – Calculator.java

```
public class Calculator {
  ...
  private void calculate(){
    if (operator == Operator.ADD){
      leftOperand = leftOperand + rightOperand;
    } else if (operator == Operator.SUBTRACT){
      leftOperand = leftOperand - rightOperand;
    } else if (operator == Operator.MULTIPLY){
      leftOperand = leftOperand * rightOperand;
    } else if (operator == Operator.DIVIDE){
      leftOperand = leftOperand / rightOperand;
    }
    rightOperand = 0;
  }

  public String getDisplayedText(){
    if (operator != null){
      return leftOperand + operator.toString() + rightOperand;
    } else {
      return "" + rightOperand;
    }
  }
}
```

Activity életciklusa

- Látható, hogy ha a programból átváltunk egy másik alkalmazásra, újraindítjuk, vagy csak elforgatjuk a képernyőt, akkor az alkalmazásunk nem őrzi meg a már bevitt adatokat.
- Az Android operációs rendszer erőforráskezelője a nem futó alkalmazásoktól megvonja azokat az erőforrásokat, amelyekre azoknak éppen nincsen szükségük.
- Az Android a program életciklusa során metódushívásokat küld, hogy tudjuk mikor kell az állapotot menteni / helyreállítani.


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Állapot mentése / helyreállítása

- Az állapotot Bundle-ban tároljuk kulcs-érték párok formájában. A kulcs String, az érték pedig elemi típus lehet csak.
- Állapot mentése
  - onSaveInstanceState metódus paramétereként kapott Bundle-ba mentjük az adatokat
- Állapot helyreállítása
  - onCreate és onRestoreInstanceState paramétereként is megkapjuk a már mentett állapotot. Előbbi a felület létezése előtt hívódik meg, míg utóbbi utána.
- Egészítsük ki a programunkat az állapotok helyes kezelésével is.

Állapot mentése / helyreállítása

```
public class MainActivity extends AppCompatActivity {
  ...
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    ...
    if (savedInstanceState != null){
      calc = new Calculator(savedInstanceState);
      txtResult.setText(calc.getDisplayedText());
    } else {
      calc = new Calculator();
    }
  }

  @Override
  protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    calc.onSaveInstanceState(outState);
  }
}
```

Állapot mentése / helyreállítása

```
public class Calculator {
  ...
  public Calculator(Bundle state) {
    leftOperand = state.getInt("leftOperand");
    rightOperand = state.getInt("rightOperand");
    operator = Operator.getOperator(state.getChar("operator"));
  }

  public void onSaveInstanceState(Bundle outState) {
    outState.putInt("leftOperand", leftOperand);
    outState.putInt("rightOperand", rightOperand);
    char op = (operator == null) ? ' ' : operator.getOpChar();
    outState.putChar("operator", op);

  }

  ...
}
```

Képernyő elforgatás

- Észrevehetjük, hogy az alkalmazásunk a képernyő elforgatás hatására újra betölti a felületet, azaz az alkalmazás elölről indul, és ismét lefut az onCreate.
- Ha azt szeretnénk, hogy a program csak álló vagy fekvő képernyős megjelenítést adhasson, akkor ezt az AndroidManifest.xml fájlban kényszeríthetjük ki az orientation megadásával.
- Gyakoribb az, hogy egy program más elrendezést használ az álló, illetve fekvő módhoz. Ezzel mi egyelőre nem foglalkozunk, de értelemszerűen két layout xml fájlt kellene készítenünk, és az onCreate-ben mindig a képernyőmódnak megfelelőt betölteni.

Képernyő forgatás zárolása

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="hurrycane.calculator">

  <application android:allowBackup="true"
               android:icon="@mipmap/ic_launcher"
               android:label="@string/app_name"
               android:roundIcon="@mipmap/ic_launcher_round"
               android:supportsRtl="true"
               android:theme="@style/AppTheme">

    <activity  android:name=".MainActivity"
               android:screenOrientation="sensorPortrait">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```