A.2 - Interfata grafica

INTERFAȚA GRAFICĂ ÎN ACTIVITY

ATRIBUT	TIP OBIECT	DESCRIERE
layout_width	View / ViewGroup	lățimea obiectului
layout_height	View / ViewGroup	înălțimea obiectului
layout_marginTop	View / ViewGroup	spațiu suplimentar ce trebuie alocat în partea de sus a obiectului
layout_marginBottom	View / ViewGroup	spațiu suplimentar ce trebuie alocat în partea de jos a obiectului
layout_marginLeft	View / ViewGroup	spațiu suplimentar ce trebuie alocat în partea din stânga a obiectului
layout_marginRight	View / ViewGroup	spațiu suplimentar ce trebuie alocat în partea din dreapta a obiectului
layout_gravity	View	modul de poziționare a elementelor componente în cadrul unui container
layout_weight	View	proporția pe care o are controlul, raportată la întregul conținut al containerului
layout_x	View / ViewGroup	poziția pe coordonata x
layout_y	View / ViewGroup	poziția pe coordonata y

Layout_width și layout_height pot să ia valorile:

- **match_parent** (ocupă tot spațiul containerului său)
- **wrap_content**

Unitate de măsură: **dp**

Controale:

1. **TextView** - afișare text care nu poate fi modificat

```
<TextView
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hello, Android!"
        android:textSize="24sp" />
```

În MainActivity poate fi folosit în metoda onCreate():

```
TextView textView = findViewById(R.id.myTextView);
textView.setText("Hello, Android!");
```

2. **EditText** - input text de la utilizator

```
<EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here" />
```

În MainActivity poate fi folosit în metoda onCreate():

```
editText = findViewById(R.id.editText);
getTextButton.setOnClickListener(v -> {
```

```
            String inputText = editText.getText().toString();
            resultTextView.setText("You entered: " + inputText);
        });
```

- Valoarea va fi afisata by default doar pe o linie
- Pentru mai multe linii: `inputType="textMultiLine"`, totodata se poate specifica si pe cate linii sa fie prin proprietatea: `lines`

3. **CheckBox** - un buton cu doua stari

```
<CheckBox
        android:id="@+id/myCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="My Checkbox" />
```

În MainActivity poate fi folosit în metoda onCreate():

```
public class MainActivity extends AppCompatActivity {

    private CheckBox checkBox;
    private TextView statusTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        checkBox = findViewById(R.id.myCheckBox);
        statusTextView = findViewById(R.id.statusTextView);

        checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {
            updateStatus(isChecked);
        });

        // Set initial status
        updateStatus(checkBox.isChecked());
    }

    private void updateStatus(boolean isChecked) {
        String status = isChecked ? "Checked" : "Unchecked";
        statusTextView.setText("Checkbox status: " + status);
    }
}
```

4. **Button**

```
<Button
```

```
        android:id="@+id/getTextButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="16dp"
        android:text="Get Text" />
```

```
PENTRU A DEZACTIVA SCRIEREA INTR-UN EDITTEXT, FOLOSIM:
android:enabled="false"
```

LAYOUTS

Cele mai utilizate tipuri de grupuri de componente vizuale sunt **`LinearLayout`**, `AbsoluteLayout`, `RelativeLayout`, `FrameLayout`, `TableLayout` și **`GridLayout`**.
Cele mai frecvent utilizate atribute sunt:

- layout_height, layout_width - definesc lățimea și înălțimea componentei, putând avea valorile:
    - `match_parent` - va ocupa tot spațiul pus la dispoziție de componenta în care este conținută (fără padding);
    - `wrap_content` - va ocupa doar spațiul solicitat de componentele pe care le conține (cu padding);
    - o valoare indicată explicit împreună cu unitatea de măsură.
- layout_weight - proporția pe care o ocupă în raport cu alte componente;
- weightSum - suma proporțiilor tuturor controalelor grafice conținute; valoarea implicită este 1;
- `layout_gravity` - modul în care componenta este aliniată în cadrul grupului din care face parte (valorile posibile pe care le poate lua această proprietate sunt: `top`, `botttom`, `left`, `right`, `center_vertical`, `center_horizontal`, `center` (centrare pe ambele direcții), `fill_vertical`, `fill_horizontal`, `fill` (ocuparea spațiului pe ambele direcții), `clip_vertical`, `clip_horizontal`; aceste valori pot fi combinate prin intermediul operatorului `|` (pe biți);

LinearLayout

- Componentele pot fi aranjate pe orizontala (default)/verticala (modificat prin proprietatea `orientation`)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
```

```
    android:padding="16dp">

    <Button
        android:id="@+id/btnMonday"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Monday"
        android:layout_marginBottom="8dp" />
</LinearLayout>
```

GridLayout

- Sub forma de tabela
- Se introduc `rowCount` si `columnCount`, trebuie specificat la inceput numarul

```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="2"
    android:rowCount="3"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"
        android:layout_column="0"
        android:layout_row="0"
        android:layout_marginEnd="8dp" />

    <EditText
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_column="1"
        android:layout_row="0"
        android:layout_gravity="fill_horizontal" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"
        android:layout_column="0"
        android:layout_row="1"
        android:layout_marginEnd="8dp" />
```

```
    <EditText
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_column="1"
        android:layout_row="1"
        android:layout_gravity="fill_horizontal" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_column="0"
        android:layout_row="2"
        android:layout_columnSpan="2"
        android:layout_gravity="center_horizontal" />

</GridLayout>
```

În cazul în care se dorește extinderea unui element grafic pe mai multe rânduri sau pe mai multe coloane, se vor utiliza atributele layout_rowSpan și layout_columnSpan. Pentru controlul modului de dispunere se va folosi proprietatea layout_gravity. Precizarea layout_width și layout_height nu este neapărat necesară, valoarea lor implicită în acest caz fiind wrap_content.

A.2. [20%] Să se creeze o interfață grafică care conține **două câmpuri text**, **needitabile**, **dispuse pe o linie**, unul lângă celălalt ocupând **jumătate** din suprafața containerului din care fac parte, având dedesubt **câte un buton** centrat relativ la ele. În cadrul câmpurilor text, se va afișa o cifră reprezentând numărul de apăsări al butonului (inițial, valoarea va fi **0**).

```
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".PracticalTest01MainActivity">

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:rowCount="2"
        android:columnCount="2"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">

        <EditText
            android:id="@+id/leftText"
            android:layout_column="0"
            android:layout_row="0"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginEnd="5dp"
            android:enabled="false"/>

        <EditText
            android:id="@+id/rightText"
            android:layout_column="1"
            android:layout_row="0"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
```

```
            android:textAlignment="center"
            android:layout_marginStart="5dp"
            android:enabled="false"/>

        <Button
            android:id="@+id/leftButton"
            android:text="Press me"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:layout_column="0"
            android:layout_row="1"
            android:layout_marginTop="20dp"/>

        <Button
            android:id="@+id/rightButton"
            android:text="Press me too"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_row="1"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:layout_marginTop="20dp"
            />
    </GridLayout>
</LinearLayout>
```

B.2. [15%] a) Forțați sistemul de operare Android **să nu salveze** / restaureze în mod automat starea câmpurilor text în momentul în care aceasta este distrusă din cauza asigurării necesarului de resurse.

```
android:saveEnabled="false"
```

ATRIBUTE PENTRU UI:

Atribute comune:

```
android:visibility="visible" - view-ul e vizibil (default)
android:visibility="invisible" - view-ul e invizibil DAR ocupă spațiu
android:visibility="gone" - view-ul e invizibil ȘI NU ocupă spațiu
android:enabled="true" - view-ul e activ și poate fi interacționat
(default)
android:enabled="false" - view-ul e dezactivat, gri, nu poate fi
interacționat
android:clickable="true" - poate fi apăsat
android:focusable="true" - poate primi focus (ex: când apeși Tab)
android:background="#FFFFFF" - culoarea de fundal
android:backgroundTint="@color/primary" - colorează fundalul (fără să
schimbi drawable-ul)
```

LinearLayout specific:

```
android:orientation="horizontal" - copiii sunt aranjați pe orizontală (pe
linie)
android:orientation="vertical" - copiii sunt aranjați pe verticală
(coloană)
android:gravity="center" - poziționează CONȚINUTUL (copiii) în centru
android:gravity="start" - conținutul la stânga (sau dreapta pentru RTL)
android:gravity="end" - conținutul la dreapta
android:gravity="top" - conținutul sus
android:gravity="center_horizontal" - centrat pe orizontală
android:layout_gravity="center" - poziționează VIEW-UL ÎNSUȘI în
părintele lui
android:weightSum="10" - suma totală pentru weight-uri (default = suma
weight-urilor copiilor)
android:baselineAligned="false" - dezactivează alinierea după
baseline-ul textului (optimizare)
android:divider="@drawable/divider" - drawable pentru separator între
copii
android:showDividers="middle" - arată separatorul între copii (nu la
capete)
```

GridLayout specific:

```
android:rowCount="2" - numărul de rânduri din grid
android:columnCount="2" - numărul de coloane din grid
android:layout_row="0" - pe ce rând e poziționat view-ul (indexare de la
0)
android:layout_column="0" - pe ce coloană e poziționat view-ul
android:layout_rowSpan="2" - view-ul ocupă 2 rânduri (merge rows)
android:layout_columnSpan="2" - view-ul ocupă 2 coloane
android:layout_rowWeight="1" - distribuie spațiul pe verticală
proporțional
android:layout_columnWeight="1" - distribuie spațiul pe orizontală
proporțional
android:layout_gravity="fill" - umple întreaga celulă
android:layout_gravity="fill_horizontal" - umple doar pe orizontală
android:alignmentMode="alignBounds" - aliniază după margini
android:columnOrderPreserved="false" - permite reordonare coloane pentru
optimizare
android:useDefaultMargins="true" - adaugă marje default între celule
```

EditText specific:

```
<!-- Text și aspect -->
android:text="Default text" - textul inițial afișat
android:hint="Enter text here" - text sugestiv când e gol (dispare la
scriere)
android:textColor="#000000" - culoarea textului
android:textColorHint="#999999" - culoarea hint-ului
android:textSize="16sp" - dimensiunea textului (sp = scale-independent
pixels)
android:textStyle="bold" - stilul: normal/bold/italic
android:textAlignment="center" - alinierea textului în view

<!-- Input și taste -->
android:inputType="text" - tip normal de text
android:inputType="textPassword" - ascunde textul (parole)
android:inputType="number" - doar cifre
android:inputType="phone" - tastatură pentru telefon
android:inputType="email" - tastatură cu @ și .com
android:maxLength="10" - maximum 10 caractere
android:maxLines="3" - maximum 3 linii de text
android:singleLine="true" - o singură linie (nu face wrap)
android:imeOptions="actionDone" - butonul de pe tastatură:
Done/Next/Search
android:digits="0123456789" - doar cifrele specificate sunt permise
```

```
<!-- Stare -->
android:cursorVisible="false" - ascunde cursorul care pâlpâie
android:selectAllOnFocus="true" - selectează tot textul când primește
focus

<!-- Design -->
android:drawableStart="@drawable/icon" - icon la început (stânga)
android:drawableEnd="@drawable/icon" - icon la final (dreapta)
android:drawablePadding="8dp" - spațiu între icon și text
```

Button specific:

```
android:textAllCaps="true" - transformă textul în majuscule (default
pentru Button)
android:textAllCaps="false" - păstrează textul cum l-ai scris tu

app:cornerRadius="8dp" - colțuri rotunjite (MaterialButton)
app:strokeColor="@color/border" - culoarea bordurii
app:strokeWidth="2dp" - grosimea bordurii
app:rippleColor="@color/ripple" - culoarea efectului de undă la apăsare

app:icon="@drawable/ic_add" - icon în buton
app:iconGravity="start" - poziția iconului:
start(stânga)/end/top/textStart
app:iconPadding="8dp" - spațiu între icon și text
app:iconTint="@color/white" - culoarea iconului

android:stateListAnimator="@null" - elimină umbra/elevația default
```

TextView specific:

```
android:lineSpacingExtra="4dp" - spațiu adițional între linii
android:lineSpacingMultiplier="1.2" - multiplicator pentru spațiul între
linii (1.2 = 120%)
android:ellipsize="end" - adaugă "..." la sfârșit când textul e prea
lung
android:ellipsize="start" - adaugă "..." la început
android:ellipsize="middle" - adaugă "..." la mijloc
android:ellipsize="marquee" - text care scrollează orizontal
android:textIsSelectable="true" - utilizatorul poate selecta textul
android:autoLink="web" - face automat link-uri din URL-uri
android:autoLink="email" - face automat link-uri din email-uri
android:autoLink="phone" - face automat link-uri din numere de telefon
```

```
android:autoLink="all" - toate tipurile de link-uri
android:linksClickable="true" - link-urile pot fi apăsate
```

ImageView specific:

```
android:src="@drawable/image" - imaginea de afișat
android:scaleType="centerCrop" - umple view-ul, croppează excesul
android:scaleType="fitCenter" - scalează imaginea să încapă, păstrează
aspect ratio
android:scaleType="fitXY" - stretch imaginea să umple complet
(distorsionează)
android:scaleType="center" - centrează fără scalare
android:adjustViewBounds="true" - ajustează dimensiunile view-ului la
aspect ratio-ul imaginii
android:tint="@color/primary" - colorează imaginea (overlay)
android:contentDescription="Image description" - descriere pentru
accesibilitate (screen readers)
```

```
Extra tips:
dp = density-independent pixels (pentru dimensiuni/marje)
sp = scale-independent pixels (pentru text, respectă setările
utilizatorului)
match_parent = ocupă tot spațiul părintelui
wrap_content = doar cât e nevoie pentru conținut
gravity = poziționează ce e ÎNĂUNTRU
layout_gravity = poziționează VIEW-UL însuși în părinte
```

Alte exercitii colocviu:

A.4. Sa se creeze o interfata grafica ce contine patru campuri text, editabile, dispus intr-o tabela 2x2. In plus, va fi inclus si un buton “Set”, centrat pe orizontala, pe o alta linie, prin intermediul caruia va fi invocata o alta activitate, din cadrul aceleiasi aplicatii Android.

```
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/setButton"
        android:text="Set"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_columnWeight="1"
        android:layout_gravity="fill"
        android:layout_marginTop="20dp"
        android:gravity="center_horizontal"
        />

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:rowCount="2"
        android:columnCount="2"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">

        <EditText
            android:id="@+id/firstText"
            android:layout_column="0"
            android:layout_row="0"
            android:text="1"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginEnd="5dp"/>

        <EditText
            android:id="@+id/secondText"
            android:layout_column="1"
            android:layout_row="0"
            android:text="2"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginStart="5dp"/>

        <EditText
            android:id="@+id/thirdText"
            android:layout_column="0"
            android:layout_row="1"
```

```
            android:text="3"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginEnd="5dp"/>

        <EditText
            android:id="@+id/fourthText"
            android:layout_column="1"
            android:layout_row="1"
            android:text="4"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginStart="5dp"/>

    </GridLayout>
</LinearLayout>
```

Sa se creeze o noua activitate, in contextul aceleiasi aplicatii Android, care sa poata fi invocata doar prin intermediul unei intentii. Interfata grafica va contine patru campuri text, needitabile, de aceasta data care vor fi dispuse asemenea celor din activitatea principala.

```
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:rowCount="3"
        android:columnCount="2"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">

        <EditText
            android:id="@+id/firstText"
            android:layout_column="0"
            android:layout_row="0"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginEnd="5dp"
```

```
            android:enabled="false"/>

        <EditText
            android:id="@+id/secondText"
            android:layout_column="1"
            android:layout_row="0"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginStart="5dp"
            android:enabled="false"/>

        <EditText
            android:id="@+id/thirdText"
            android:layout_column="0"
            android:layout_row="1"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginEnd="5dp"
            android:enabled="false"/>

        <EditText
            android:id="@+id/fourthText"
            android:layout_column="1"
            android:layout_row="1"
            android:text="0"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:textAlignment="center"
            android:layout_marginStart="5dp"
            android:enabled="false"/>

        <Button
            android:id="@+id/sumButton"
            android:text="Sum"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:layout_column="0"
            android:layout_row="2"
            android:layout_marginTop="20dp"/>

        <Button
            android:id="@+id/productButton"
            android:text="Product"
            android:layout_width="0dp"
```

```
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_row="2"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:layout_marginTop="20dp"
            />

    </GridLayout>
</LinearLayout>
```

TIPS SPACE:

```
<Space
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

    <EditText
        android:id="@+id/stepText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="1"
        android:textAlignment="center"/>
```

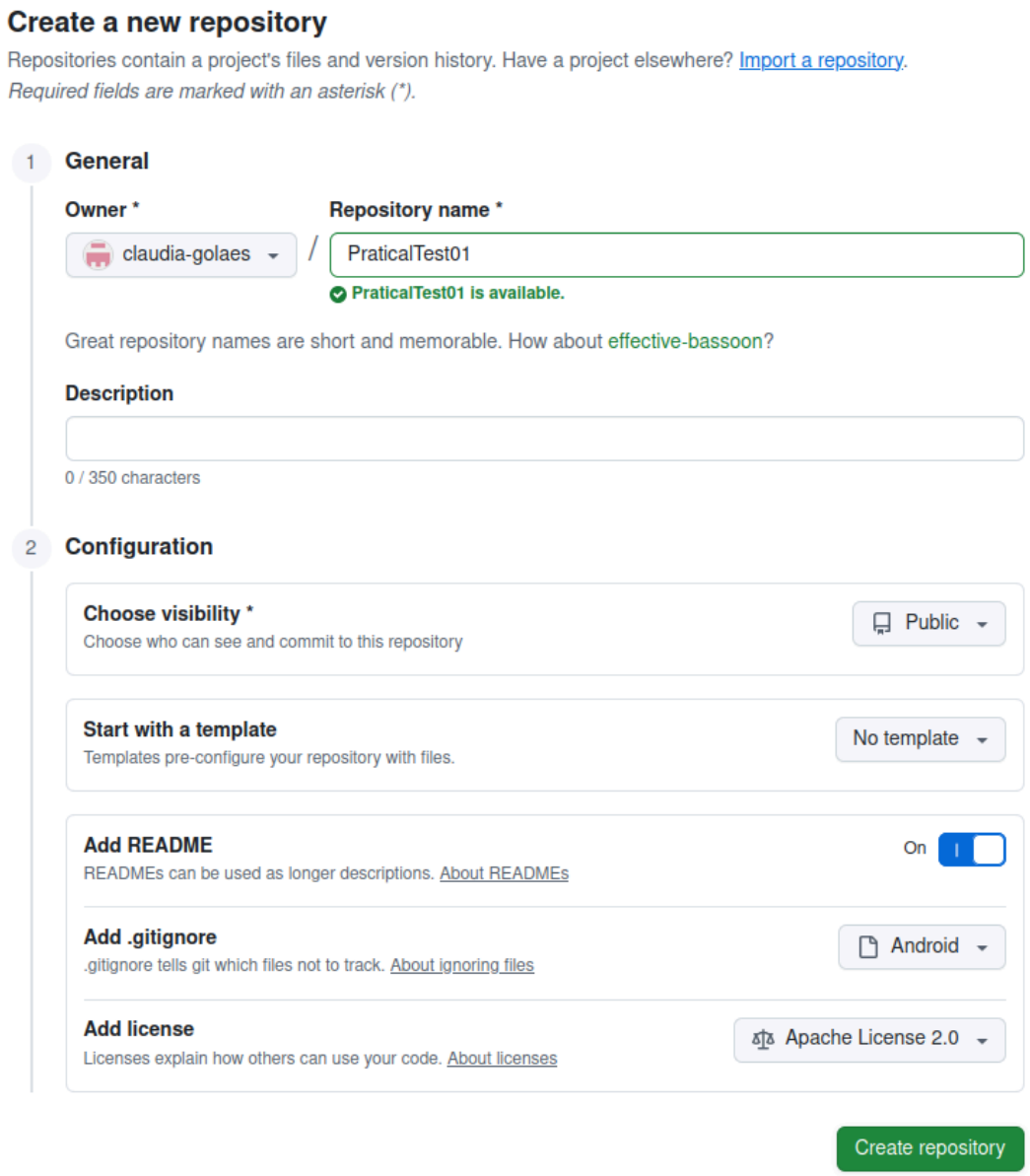
TIPS JUMATATE:

```
android:layout_width="0dp" (în loc de wrap_content)
android:layout_columnWeight="1" (la fel ca celelalte elemente)
android:layout_gravity="fill_horizontal|center_vertical"
```

A.1 - Setup

A.1. [5%] a) În contul Github personal, să se creeze un depozit denumit PracticalTest01. Inițial, acesta trebuie să conțină:

- un fișier **README.md**, în care veți preciza următoarele informații: numele și prenumele, grupadin care faceți parte;
- un fișier **.gitignore** prin intermediul căruia resursele generate ale aplicației Android (clase, fișiere .ap_, .apk, .dex) nu vor fi consemnate în cadrul sistemului de versionare a codului sursă;
- un fișier **LICENSE** care să descrie condițiile de utilizare a aplicației Android (la alegere).



b) Să se descarce pe discul local conținutul depozitului (la distanță) PracticalTest01.
git clone https://www.github.com/claudia-golaes/PracticalTest01.git

c) În mediul integrat de dezvoltare Android Studio, să se creeze un proiect corespunzător unei aplicații Android, având următoarea specificație:

- denumirea aplicației Android și a proiectului este PracticalTest01;
- denumirea pachetului corespunzător aplicației Android este ro.pub.cs.systems.eim.practicaltest01;
- nivelul minim de API este cel corespunzător versiunii Jelly Bean (4.1) = 16;

Explicație pas cu pas pentru crearea proiectului (Empty Views Activity):
https://eim.pages.upb.ro/lab1/first_app.html

- denumirea activității principale a aplicației Android este PracticalTest01MainActivity;

Pași:

- Click dreapta pe MainActivity.kt
- Refactor
- Schimbă numele în PracticalTest01MainActivity
- În PracticalTest01MainActivity.kt schmbă numele clasei
- În AndoridManifest.xml, modifică linia `android:name=".MainActivity"`

```
-   android:name=".PracticalTest01MainActivity"
```

- denumirea fișierului ce descrie interfața grafică a activității principale a aplicației Android este activity_practical_test01_main.xml.

Pași:

- În folderul layout
- Refactor fișierul activity_main.xml
- Schimbă numele în activity_practical_test01_main

B.1 - Button Listener

B.1. [10%] Să se asocieze butoanelor un **ascultător**, astfel încât pentru fiecare eveniment de tip apăsare a butonului, valoarea din câmpul text corespunzător să fie **incrementată**.

PAȘI

1) Initializarea butoanelor si a textelor in PracticalTest01MainActivity

```
private lateinit var leftEditText : EditText
private lateinit var rightEditText : EditText
private lateinit var leftButton : Button
private lateinit var rightButton : Button
```

2) Definirea unei noi clase, în care suprascriem metoda onClick() în funcție de cerințe

```
inner class ButtonListener : View.OnClickListener {
        override fun onClick(v: View?) {
            if (v?.id == R.id.left_button) {
               if (leftEditText.text.toString().equals("0")){
                    leftEditText.setText(1.toString())
               }
               else {
                    val leftValue =
Integer.valueOf(leftEditText.text.toString())
                    leftEditText.setText((leftValue + 1).toString())
               }
            }

            else if (v?.id == R.id.right_button) {
                if (rightEditText.text.toString().equals("0")){
                    rightEditText.setText(1.toString())
                }
                else {
                    val rightValue =
Integer.valueOf(rightEditText.text.toString())
                    rightEditText.setText((rightValue + 1).toString())
               }
            }
}
```

3) Definirea butoanelor si a textelor in metoda onCreate()

```
leftEditText = findViewById(R.id.leftEditText)
rightEditText = findViewById(R.id.rightEditText)
leftButton = findViewById(R.id.left_button)
rightButton = findViewById(R.id.right_button)

leftButton.setOnClickListener(ButtonListener())
rightButton.setOnClickListener(ButtonListener())
```

Exercitii:
Sa se asocieze butonului “Set” un ascultator, astfel incat, daca toate campurile contin numere, se va emite intentia care porneste a doua activitate, transmitandu-se prin intermediul acesteia valorile continute de campuri. Daca cel putin un camp nu contine un numar, actiunea click o sa fie ignorata.

```
inner class ButtonListener : View.OnClickListener {
        override fun onClick(v: View?) {
            if(v?.id == R.id.setButton){
                var firstValue = firstText.text.toString().toIntOrNull()
                var secondValue =
secondText.text.toString().toIntOrNull()
                var thirdValue = thirdText.text.toString().toIntOrNull()
                var fourthValue =
fourthText.text.toString().toIntOrNull()
                val intent = Intent(this@MainActivity,
SecondActivity::class.java).apply{
                    putExtra("firstValue", firstValue)
                    putExtra("secondValue", secondValue)
                    putExtra("thirdValue", thirdValue)
                    putExtra("fourthValue", fourthValue)
                }
                if(firstValue != null && secondValue != null &&
thirdValue != null && fourthValue != null) {
                    startForResult.launch(intent)
                }
            }
        }
    }
```

C - Intent

Pași

1) In activitatea main, se intializeaza ActivityResultLauncher

```
    private lateinit var startForResult: ActivityResultLauncher<Intent>
```

2) In activitatea main, se construieste pe baza urmatorului schelete

```
startForResult = registerForActivityResult(StartActivityForResult()) { result ->
            if (result.resultCode == Activity.RESULT_OK) {
                // Procesează rezultatul aici
                val data: Intent? = result.data
                val someData = data?.getStringExtra("another_key")
                // Folosește someData cum este necesar si aici este ce primeste de la
copil
            }
        }
        val btn = findViewById<Button>(R.id.open_activity_button)
        btn.setOnClickListener {
            val intent = Intent(this, SecondActivity::class.java).apply {
                putExtra("some_key", "someValue")
            }
            // Lansează activitatea copil folosind launcher-ul
            startForResult.launch(intent)
        }
```

3) In activitatea copil, se construieste pe baza urmatorului schelete:

```
class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.second_activity)

        // Presupunând că vrei să returnezi rezultatul la un anumit eveniment, de
exemplu, la apăsarea unui buton
        val someButton: Button = findViewById(R.id.some_button)
        someButton.setOnClickListener {
            val returnIntent = Intent().apply {
                putExtra("another_key", "anotherValue")
            }
            setResult(RESULT_OK, returnIntent)
            finish()
        }

        // Dacă activitatea se încheie fără a seta explicit un rezultat, poți să nu faci
nimic sau să setezi RESULT_CANCELED
    }
}
```

4) Pentru a prelua datele in activitatea copil:

```
        val intent = getIntent()
        val data = intent.extras
```

TOAST:

```
val text = "Hello toast!"
val duration = Toast.LENGTH_LONG

val toast = Toast.makeText(this, text, duration) // in Activity
toast.show()
```

Exercitii colocviu:
Sa se asocieze butonului “Set” un ascultator, astfel incat, daca toate campurile contin numere, se va emite intentia care porneste a doua activitate, transmitandu-se prin intermediul acesteia valorile continute de campuri. Daca cel putin un camp nu contine un numar, actiunea click o sa fie ignorata. In activitatea principala salvati Suma si Produsul in variabile neasociate cu interfata grafica, si sunt afisate in **Toast**

1. **In MainActivity.kt**

```
inner class ButtonListener : View.OnClickListener {
        override fun onClick(v: View?) {
            if(v?.id == R.id.setButton){
                var firstValue = firstText.text.toString().toIntOrNull()
                var secondValue = secondText.text.toString().toIntOrNull()
                var thirdValue = thirdText.text.toString().toIntOrNull()
                var fourthValue = fourthText.text.toString().toIntOrNull()
                val intent = Intent(this@MainActivity,
SecondActivity::class.java).apply{
                    putExtra("firstValue", firstValue)
                    putExtra("secondValue", secondValue)
                    putExtra("thirdValue", thirdValue)
                    putExtra("fourthValue", fourthValue)
                }
                if(firstValue != null && secondValue != null && thirdValue != null &&
fourthValue != null) {
                    startForResult.launch(intent)
                }
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }

        startForResult =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
            if (result.resultCode == Activity.RESULT_OK) {
                val data: Intent? = result.data
                val sum = data?.getStringExtra("sum")
                val product = data?.getStringExtra("product")
                if(sum != null) {
                    val duration = Toast.LENGTH_LONG
                    val text = "Sum is $sum"
                    val toast = Toast.makeText(this, text, duration)
                    toast.show()
                }
                if(product != null){
                    val duration = Toast.LENGTH_LONG
                    val text = "Product is $product"
                    val toast = Toast.makeText(this, text, duration)
                    toast.show()
                }
            }
        }
```

```
        firstText = findViewById(R.id.firstText)
        secondText = findViewById(R.id.secondText)
        thirdText = findViewById(R.id.thirdText)
        fourthText = findViewById(R.id.fourthText)
        setButton = findViewById(R.id.setButton)

        setButton.setOnClickListener(ButtonListener())
    }
```

2. In SecondActivity

```
inner class ButtonListener : View.OnClickListener{
        override fun onClick(v: View?) {
            if(v?.id == R.id.sumButton) {
                val returnIntent = Intent().apply {
                    var firstValue = firstText.text.toString().toInt()
                    var secondValue = secondText.text.toString().toInt()
                    var thirdValue = thirdText.text.toString().toInt()
                    var fourthValue = fourthText.text.toString().toInt()
                    var sum = (firstValue + secondValue + thirdValue +
fourthValue).toString()
                    putExtra("sum", sum)
                }
                setResult(RESULT_OK, returnIntent)
                finish()
            }

            if(v?.id == R.id.productButton) {
                val returnIntent = Intent().apply {
                    var firstValue = firstText.text.toString().toInt()
                    var secondValue = secondText.text.toString().toInt()
                    var thirdValue = thirdText.text.toString().toInt()
                    var fourthValue = fourthText.text.toString().toInt()
                    var product = (firstValue * secondValue * thirdValue *
fourthValue).toString()
                    putExtra("product", product)
                }
                setResult(RESULT_OK, returnIntent)
                finish()
            }
        }

    }


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_second)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }

        firstText = findViewById(R.id.firstText)
        secondText = findViewById(R.id.secondText)
        thirdText = findViewById(R.id.thirdText)
        fourthText = findViewById(R.id.fourthText)
        sumButton = findViewById(R.id.sumButton)
        productButton = findViewById(R.id.productButton)
```

```
        sumButton.setOnClickListener(ButtonListener())
        productButton.setOnClickListener(ButtonListener())

        val intent = getIntent()
        val data = intent.extras

        if(data!= null) {
            val firstValue = data.getInt("firstValue")
            val secondValue = data.getInt("secondValue")
            val thirdValue = data.getInt("thirdValue")
            val fourthValue = data.getInt("fourthValue")

            firstText.setText(firstValue.toString())
            secondText.setText(secondValue.toString())
            thirdText.setText(thirdValue.toString())
            fourthText.setText(fourthValue.toString())
        }
```

Restaurarea starii

Se da override la functia onSave InstanceState():

```
override fun onSaveInstanceState(savedInstanceState: Bundle) {
    /* Trebuie sa apelam metoda din clasa de baza întrucât API-ul Android
    furnizează o implementare implicită pentru salvarea stării unei
    activități, parcurgând ierarhia de componente grafice (obiecte de tip
    View) care au asociat un identificator (android:id), folosit drept
    cheie în obiectul Bundle. Astfel, de regulă, pentru elementele
    interfeței grafice, nu este necesar să se mențină starea, acest lucru
    fiind realizat în mod automat, cu respectarea condiției menționate.
    super.onSaveInstanceState(savedInstanceState); */

    super.onSaveInstanceState(savedInstanceState)

    /* Determian o referinta pentru obiectul de tip EditText din interfata grafica
       cu ID-ul username_edit_text */
    val usernameEditText = findViewById(R.id.username_edit_text) // <tipul> in caz de
eroarea: Cannot infer type for this parameter. Specify it explicitly.
Not enough information to infer type argument for 'T'.
    savedInstanceState.putString("SOME_STRING_USED_AS_KEY",
usernameEditText.text.toString())
}
```

Dupa aia in MainActivity.kt, in metoda OnCreate() se restaureaza starea

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_lifecycle_monitor);
      EditText usernameEditText = (EditText)findViewById(R.id.username_edit_text);
      if ((savedInstanceState != null) &&
(savedInstanceState.getString(Constants.USERNAME_EDIT_TEXT) != null)) {

usernameEditText.setText(savedInstanceState.getString(Constants.USERNAME_EDIT_TEXT));
      }
    }
```

Exercitiu colocviu:
C.2.b) Implementati un mecanism pentru salvarea si restaurarea acestor variabile, pentru situatia in care sistemul de operare Android distruge activitate:

1) Am facut variabille globale:

```
    private var sum: Int = -1
    private var product: Int = -1
```

2) Override onSaveInstanceState()

```
override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        if (sum != 0) {
            outState.putInt("sum", sum)
        }
        if (product != 0) {
            outState.putInt("product", product)
        }
```

```
    }
```

3) In metoda onCreate():

```
if (savedInstanceState != null) {
            sum = savedInstanceState.getInt("sum")
            Log.w("RESTORE", "The sum is $sum")
            product = savedInstanceState.getInt("product")
            Log.w("RESTORE", "The product is $product")
        }
```

D. Servicii

Pasi Servicii:

1) Declararea serviciului in AndroidManifest.xml, cu tagul service, in cadrul elementului application.

EX:

```
<manifest ...>
  <application ...>
    <service
      android:name="ro.pub.cs.systems.eim.lab05.SomeService"
      android:enabled="true"
      android:exported="true"
      android:permission="ro.pub.cs.systems.eim.lab05.SOME_SERVICE_PERMISSION" />
  </application>
</manifest>
```

2) Variabila globala pentru a ne asigura ca se porneste un singur thread

```
private var serviceStarted = false
```

3) In metoda onCreate() se proneste un serviciu:

```
val intent = Intent(this, SomeService::class..java)
startService(intent)
serviceStarted = true
```

4) In MainService.kt se defineste clasa ProcessingThread, care primeste ca parametri obligatoriu contextul + optional argumentele care trebuie date mai departe in broadcastReceiver

```
class ProcessingThread(val context : Context) : Thread() {
    companion object {
        var isRunning = false
    }

    override fun run() {
        while(isRunning) {
            // de completat cu ce trimiterea unui broadcast

            val intent = Intent(Constants.ACTION).apply {
                setPackage(context.packageName) // !!!FOARTE IMPORTANT
            }
            Log.d("SERVICE:", "MESSAGE SENT")
            context.sendBroadcast(intent)
            sleepThread()
        }
    }

    fun sleepThread() {
        sleep(10000)
    }
    fun stopThread() {
        isRunning = false
    }
}
```

5) In MainService.kt, definim clasa MainService:

```
class MainService : Service() {

    private lateinit var processingThread : ProcessingThread

    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    override fun onCreate() {

    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        Log.d("SERVICE:", "STARTING SERVICE")
        processingThread = ProcessingThread(this@MyService)
        if(ProcessingThread.isRunning == false){
            ProcessingThread.isRunning = true
            processingThread.start()
        }
        return START_NOT_STICKY
    }

    override fun onDestroy() {
        Log.d("SERVICE:", "STOPPING THREAD")
        processingThread.stopThread()
    }
}
```

6) Pentru a defini action-ul pe care il folosim in trimiterea broadcastului, in folderul in care avem si activitatile: Click dreapta —> New —> Kotlin/Class File—>Object—>Constants.kt

```
package ro.pub.cs.systems.eim.practicaltest01

object Constants {
    const val ACTION = "ro.pub.cs.systems.eim.practicaltest01.ACTION"
}
```

7) Pentru a primi broadcastul se face un nou fisier de tip BroadcastReceiver, iar clasa poate primi optional ca parametri anumite field-uri din activitate pe care sa le modifice odata cu instantierea clasei

```
package ro.pub.cs.systems.eim.practicaltest01
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.util.Log
class PracticalTest01BroadcastReceiver : BroadcastReceiver() {
override fun onReceive(context: Context, intent: Intent) {
    Log.d("BROADCAST RECEIVER", "BROADCAST RECEIVER GOT MESSAGE")
    val data = intent.getExtras()
    if (data != null) {

    Log.d("BROADCAST RECEIVER", "Received broadcast with action
    $action ")
    }
}
```

8) In MainActivity.kt definim BroadcastReceiverul:

```
private lateinit var broadcastReceiver : PracticalTest01BroadcastReceiver
```

9) In metoda onCreate() din MainActivity.kt il intializam:

```
broadcastReceiver = PracticalTest01BroadcastReceiver() // + ca argumente optionale
```

10) In MainActivity.kt oprim serviciul: **DACA SE MENTIONEAZA** ca serviciul se opreste atunci cand activitatea principala este distrusa.

```
override fun onDestroy() {
        if (serviceStarted == true) {
            val intent = Intent(
                this@PracticalTest01Var07MainActivity,
                PracticalTest01Var07Service::class.java
            )
            stopService(intent)
        }
        super.onDestroy()
    }
```

11) In MainActivity.kt suprascriem metoda onResume():

```
override fun onResume() {
        super.onResume()
        val intentFilter = IntentFilter().apply {
            addAction(Constants.ACTION) // aici adaugam toate actiunile
        }

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            registerReceiver(broadcastReceiver, intentFilter,
Context.RECEIVER_NOT_EXPORTED)
        } else {
            registerReceiver(broadcastReceiver, intentFilter)
        }
        Log.d("[MAIN]", "BroadcastReceiver registered!")
    }
```

```
    @SuppressLint("UnspecifiedRegisterReceiverFlag") // Functia
onResume() trebuie sa aiba acest tag
```

12) Suprascriem metoda onPause()

```
    override fun onPause() {
        super.onPause()
        unregisterReceiver(broadcastReceiver)
    }
```

Exemplu fisier AndroidManfiest.xml (cand BroadcastReceiver este facut drept clasa)

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PracticalTest01">
        <receiver
            android:name=".PracticalTest01Var07BroadcastReceiver"
            android:enabled="true"
            android:exported="true"></receiver>

        <service
            android:name=".PracticalTest01Var07Service"
            android:enabled="true"
            android:exported="true" />

        <activity
            android:name=".PracticalTest01Var07SecondaryActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.Secondary" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".PracticalTest01Var07MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Exemplu BroadcastReceiver: Sa se implementeze un ascultator pentru mesajele de difuzare. Procesarea unui mesaj cu difuzare implica suprascrierea campurilor text cu valorile primite/

```
package ro.pub.cs.systems.eim.practicaltest01

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.util.Log
import android.widget.EditText

class PracticalTest01Var07BroadcastReceiver(val firstEditText : EditText, val
secondEditText : EditText, val thirdEditText : EditText, val fourthEditText : EditText)
: BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        val data = intent.extras
        val value1 = (data?.getInt("NUMBER1").toString())
        val value2 = (data?.getInt("NUMBER2").toString())
        val value3 = data?.getInt("NUMBER3").toString()
        val value4 = data?.getInt("NUMBER4").toString()
        // This method is called when the BroadcastReceiver is receiving an Intent
broadcast.
        Log.d("BROADCAST RECEIVER", "BROADCAST RECEIVER GOT MESSAGE WITH DATA: $value1,
$value2, $value3, $value4")

        if (data != null) {
            firstEditText.setText(data.getInt("NUMBER1").toString())
            secondEditText.setText(data.getInt("NUMBER2").toString())
            thirdEditText.setText(data.getInt("NUMBER3").toString())
            fourthEditText.setText(data.getInt("NUMBER4").toString())
        }
    }
}
```

Exemplu BroadcastReceiver: Sa se implementeze un ascultator pentru mesajele de difuzare. Procesarea unui mesaj cu difuzare implica jurnaliizarea sa in consola, folosind o anumita eticheta.

```
package ro.pub.cs.systems.eim.practicaltest01
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.util.Log
class PracticalTest01BroadcastReceiver : BroadcastReceiver() {
override fun onReceive(context: Context, intent: Intent) {
Log.d("BROADCAST RECEIVER", "BROADCAST RECEIVER GOT MESSAGE")// This method is called
when the BroadcastReceiver is receiving an Intent broadcast.
val data = intent.getExtras()
if (data != null) {
val date = data.getString("DATE")
val arithmetic = data.getDouble("ARITHMETIC_MEAN", Double.NaN)
val geometric = data.getDouble("GEOMETRIC_MEAN", Double.NaN)
val action = data.getInt("ACTION", -1)
Log.d("BROADCAST RECEIVER", "Received broadcast with action
```

```
$action at $date: Arithmetic Mean = $arithmetic, Geometric Mean =
$geometric")
}
}
}
```

Exemplu Serviciu colocviu: Sa se implementeze un serviciu **started** care este pornit de activitatea principala si propaga la nivelul sistemului de operare Andorid un mesaj cu difuzare, la fiecare 10 secunde, ce contine patru valori intregi aleatorii cu care se vor suprascrie in activitatea principala campurile text.

```
package ro.pub.cs.systems.eim.practicaltest01

import android.app.Service
import android.content.Context
import android.content.Intent
import android.os.IBinder
import android.util.Log
import kotlin.random.Random

class ProcessingThread(val context : Context) : Thread() {
    companion object {
        var isRunning = false
    }

    override fun run() {
        while(isRunning) {
            val random1 = Random.nextInt(0, 100)
            val random2 = Random.nextInt(0, 100)
            val random3 = Random.nextInt(0, 100)
            val random4 = Random.nextInt(0, 100)

            val intent = Intent(Constants.ACTION).apply {
                setPackage(context.packageName)
                putExtra("NUMBER1", random1)
                putExtra("NUMBER2", random2)
                putExtra("NUMBER3", random3)
                putExtra("NUMBER4", random4)
            }
            Log.d("SERVICE:", "MESSAGE SENT")
            context.sendBroadcast(intent)
            sleepThread()
        }
    }

    fun sleepThread() {
        sleep(10000)
    }
    fun stopThread() {
        isRunning = false
    }
}
class PracticalTest01Var07Service : Service() {

    private lateinit var processingThread : ProcessingThread
```

```
    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    override fun onCreate() {

    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        Log.d("SERVICE:", "STARTING SERVICE")
        processingThread = ProcessingThread(this@PracticalTest01Var07Service)
        if(ProcessingThread.isRunning == false){
            ProcessingThread.isRunning = true
            processingThread.start()
        }
        return START_NOT_STICKY
    }

    override fun onDestroy() {
        Log.d("SERVICE:", "STOPPING THREAD")
        processingThread.stopThread()
    }
}
```

Exemplu Serviciu colocviu: Sa se implementeze un serviciu de tip started care primeste prin intentia cu care e invocat numerele din campurile text si propaga la nivelul sistemului de operare Android un mesaj cu difuzare la fiecare 10 secunde ce cntine data si ora la care a fost transmis, mediar aritmetica si media geometrica a numerele. Definiti 3 actiuni diferite, care vor fi asociate aleator intentiei folosite pentru propagarea mesajului cu difuzare.

```
package ro.pub.cs.systems.eim.practicaltest01
import android.app.Service
import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.icu.util.Calendar
import android.os.Build
import android.os.Handler
import android.os.IBinder
import android.os.Looper
import android.os.Message
import android.util.Log
import androidx.annotation.RequiresApi
import java.time.LocalDate
import java.time.LocalDateTime
import java.time.format.DateTimeFormatter
class PracticalTest01Service : Service() {
private lateinit var processingThread : ProcessingThread
companion object {
const val ACTION1 = "ro.pub.cs.systems.eim.practicaltest01.SomeAction.ACTION1"
const val ACTION2 = "ro.pub.cs.systems.eim.practicaltest01.SomeAction.ACTION2"
const val ACTION3 = "ro.pub.cs.systems.eim.practicaltest01.SomeAction.ACTION3"
}
class ProcessingThread(private val context : Context, private val
```

```
leftValue : Int, private val rightValue : Int) : Thread() {
var isRunning = true
private val arithmeticMean = (leftValue + rightValue) / 2.0
private val geometricMean = Math.sqrt((leftValue *rightValue).toDouble())
private val formatter = SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
override fun run() {
Log.d("[SERVICE]", "Starting service thread...")
while (isRunning) {
try {
val random = (1..3).random()
val time = Calendar.getInstance().time
val current_date = formatter.format(time)
val action = when(random) {
1 -> ACTION1
2 -> ACTION2
else -> ACTION3
}
val intent = Intent(action).apply {
// ADAUGĂ ASTA - face intent-ul explicit!
setPackage(context.packageName)
putExtra("DATE", current_date)
putExtra("ARITHMETIC_MEAN", arithmeticMean)
putExtra("GEOMETRIC_MEAN", geometricMean)
putExtra("ACTION", random)
}
context.sendBroadcast(intent)
Log.d("[SERVICE]", "Message sent with action $action!")
sleep(10000)
} catch (e: InterruptedException) {
break
}
}
}
fun stopThread() {
isRunning = false}
}
override fun onBind(intent: Intent): IBinder? {
return null
}
override fun onCreate() {
//
super.onCreate()
}
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
Log.d("[SERVICE]", "Starting service...")
if (intent != null) {
val data = intent.getExtras()
if (data != null) {
val leftValue = data.getInt("LEFT_VALUE")
val rightValue = data.getInt("RIGHT_VALUE")
processingThread = ProcessingThread(this, leftValue,rightValue)
processingThread.start()
}
}
return START_NOT_STICKY
}
```

```
override fun onDestroy() {
Log.d("[SERVICE]", "Service stopped!")
processingThread.stopThread()
}
}
```

Adb comenzi

Comenzi ADB Utile

Comenzi pentru a afla adresele IP

1. Obținerea tuturor adreselor IP ale dispozitivului

adb shell ip addr show

Afișează toate interfețele de rețea și adresele IP asociate (WiFi, mobile data, localhost).

2. Doar adresa IP WiFi

adb shell ip addr show wlan0

Afișează doar informații despre interfața WiFi (wlan0).

3. Adresa IP WiFi - variantă simplificată

adb shell ip -f inet addr show wlan0

Filtrează doar adresele IPv4 pentru interfața WiFi.

4. Alternativă cu ifconfig (pe dispozitive mai vechi)

adb shell ifconfig wlan0

5. Extragerea rapidă doar a IP-ului WiFi

adb shell ip route | grep wlan0

6. Verificare conexiune și IP cu netcfg (pe Android mai vechi)

adb shell netcfg

Comenzi de bază ADB

Conexiune și dispozitive

Listează dispozitivele conectate
adb devices

Listează dispozitivele cu detalii suplimentare

```
adb devices -l

# Conectare la dispozitiv prin IP (după ce ai aflat IP-ul)
adb connect <IP_ADDRESS>:5555

# Deconectare de la dispozitiv
adb disconnect <IP_ADDRESS>:5555

# Începe serverul ADB
adb start-server

# Oprește serverul ADB
adb kill-server

# Selectează un dispozitiv specific (când ai mai multe conectate)
adb -s <device_id> <command>
```

Instalare și gestionare aplicații

```
# Instalează o aplicație APK
adb install app.apk

# Instalează APK suprascriind aplicația existentă
adb install -r app.apk

# Dezinstalează o aplicație
adb uninstall com.example.app

# Listează toate pachetele instalate
adb shell pm list packages

# Listează doar pachetele third-party
adb shell pm list packages -3

# Găsește calea unei aplicații instalate
adb shell pm path com.example.app

# Extrage un APK de pe dispozitiv
adb pull /data/app/com.example.app-1/base.apk
```

Transferul de fișiere

```
# Copiază fișier de pe PC pe dispozitiv
adb push local_file.txt /sdcard/
```

```
# Copiază fișier de pe dispozitiv pe PC
adb pull /sdcard/file.txt ./

# Listează conținutul unui director de pe dispozitiv
adb shell ls /sdcard/
```

Loguri și debugging

```
# Afișează logurile în timp real
adb logcat

# Filtrează logurile după tag
adb logcat -s TAG_NAME

# Șterge logurile
adb logcat -c

# Salvează logurile într-un fișier
adb logcat > logs.txt

# Loguri doar pentru o aplicație specifică
adb logcat | grep com.example.app

# Loguri cu filtrare după prioritate (V-verbose, D-debug, I-info, W-warning, E-error, F-fatal)
adb logcat *:E
```

Shell și comenzi pe dispozitiv

```
# Deschide shell interactiv
adb shell

# Execută o comandă directă
adb shell <command>

# Obține informații despre dispozitiv
adb shell getprop

# Obține versiunea Android
adb shell getprop ro.build.version.release

# Obține modelul dispozitivului
adb shell getprop ro.product.model

# Obține informații despre baterie
adb shell dumpsys battery
```

Informații despre memorie
adb shell dumpsys meminfo

Informații despre CPU
adb shell dumpsys cpuinfo

Control aplicații

Pornește o aplicație/activity
adb shell am start -n com.example.app/.MainActivity

Oprește o aplicație forțat
adb shell am force-stop com.example.app

Trimite un broadcast
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED

Șterge datele aplicației
adb shell pm clear com.example.app

Screenshot și screencord

Ia screenshot și salvează pe dispozitiv
adb shell screencap /sdcard/screenshot.png

Ia screenshot și salvează direct pe PC
adb shell screencap -p /sdcard/screenshot.png
adb pull /sdcard/screenshot.png

Înregistrează ecranul (max 180 secunde)
adb shell screenrecord /sdcard/demo.mp4

Înregistrează cu timp limitat (secunde)
adb shell screenrecord --time-limit 30 /sdcard/demo.mp4

Reboot și control sistem

Reboot normal
adb reboot

Reboot în recovery mode
adb reboot recovery

```
# Reboot în bootloader/fastboot
adb reboot bootloader

# Oprește dispozitivul (necesită root pe majoritatea dispozitivelor)
adb shell reboot -p
```

Input și automatizare

```
# Simulează apăsare pe ecran (coordonate x y)
adb shell input tap 500 1000

# Simulează swipe (x1 y1 x2 y2 duration_ms)
adb shell input swipe 300 500 300 1000 100

# Introduce text
adb shell input text "Hello%sWorld"  # %s pentru spațiu

# Apasă tasta BACK
adb shell input keyevent 4

# Apasă tasta HOME
adb shell input keyevent 3

# Apasă tasta POWER
adb shell input keyevent 26

# Apasă tasta MENU
adb shell input keyevent 82

# Apasă tasta VOLUME UP/DOWN
adb shell input keyevent 24  # Volume Up
adb shell input keyevent 25  # Volume Down
```

Port forwarding

```
# Forward port de pe dispozitiv către PC
adb forward tcp:8080 tcp:8080

# Listează toate forward-urile active
adb forward --list

# Elimină un forward specific
adb forward --remove tcp:8080

# Elimină toate forward-urile
```

```
adb forward --remove-all

# Reverse port forwarding (de pe PC către dispozitiv)
adb reverse tcp:8080 tcp:8080
```

WiFi ADB (pentru debugging wireless)

```
# 1. Conectează dispozitivul prin USB
# 2. Asigură-te că dispozitivul e pe aceeași rețea WiFi

# Pornește ADB pe TCP port 5555
adb tcpip 5555

# 3. Deconectează USB-ul

# 4. Află IP-ul dispozitivului (folosind comenzile de la început)
adb shell ip addr show wlan0

# 5. Conectează-te wireless
adb connect <IP_ADDRESS>:5555

# Pentru a reveni la USB
adb usb
```

Informații despre sistem și hardware

```
# Toate proprietățile sistemului
adb shell getprop

# Rezoluția ecranului
adb shell wm size

# Densitatea ecranului
adb shell wm density

# Informații despre CPU
adb shell cat /proc/cpuinfo

# Informații despre memorie
adb shell cat /proc/meminfo

# Spațiu disponibil
adb shell df

# Lista proceselor active
```

```
adb shell ps

# Informații detaliate despre un proces
adb shell dumpsys package com.example.app
```

Comenzi utile pentru debugging Android Studio

```
# Listează emulatorii disponibili
emulator -list-avds

# Pornește un emulator specific
emulator -avd <AVD_NAME>

# Pornește emulator cu date șterse
emulator -avd <AVD_NAME> -wipe-data

# Verifică ADB merge corect cu Gradle
./gradlew installDebug

# Rulează testele pe dispozitiv conectat
./gradlew connectedAndroidTest
```

Tips și best practices

Multiple dispozitive: Folosește -s pentru a specifica dispozitivul când ai mai multe conectate:

```
adb -s emulator-5554 install app.apk
```

1.
2. Debugging wireless: Foarte util când dezvolți și nu vrei să fii legat de USB tot timpul.

Logcat cu filtre: Folosește filtre pentru a nu te pierde în loguri:

```
adb logcat ActivityManager:I MyApp:D *:S
```

3.

Backup/Restore (pentru debugging):

```
# Backup
adb backup -f backup.ab -apk -all
```

```
# Restore
adb restore backup.ab
```

4.

Performance monitoring:

```
 # FPS și frame timing
adb shell dumpsys gfxinfo com.example.app
```