


Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz				 <b>POLITECHNIKA BYDGOSKA</b> Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Przedmiot	Programowanie urządzeń mobilnych			Kierunek/Tryb	IS/ST
Nr laboratorium	2	Data wykonania	17.03.2025	Grupa	1
Ocena		Data oddania	24.03.2025	Imię Nazwisko	Cezary Tytko
Nazwa ćwiczenia	Cykl życia aktywności				

## Cel ćwiczenia laboratoryjnego

Celem ćwiczenia laboratoryjnego jest zapoznanie się z cyklem życia aktywności w aplikacjach mobilnych na platformie Android. Studenci mają za zadanie stworzyć aplikację, która będzie zarządzać stanem aktywności i wyświetlać historię zmian stanu na ekranie oraz generować logi w LogCat. Ćwiczenie ma na celu zrozumienie, jak różne metody cyklu życia aktywności wpływają na działanie aplikacji oraz jak można efektywnie zarządzać zasobami w trakcie jej działania.

## Opis projektu

Zadaniem, które dobrze obrazuje mechanizmy działania aplikacji na Androida, jest prosty **licznik kliknięć**. Aplikacja licznika kliknięć będzie miała przycisk, który po kliknięciu zwiększa licznik, oraz TextView, który wyświetla jego aktualną wartość. Wartość licznika zostanie zachowana podczas zmiany orientacji ekranu, a także po zamknięciu aplikacji.

## Implementacja

Kod aplikacji został napisany w języku Kotlin i środowisku Android Studio.

### MainActivity.kt

```

1. class MainActivity : AppCompatActivity() {
2.     private val TAG = "MainActivity"
3.     private val PREFS_NAME = "counterPrefs" // Nazwa pliku z preferencjami
4.     private val COUNTER_KEY = "clickCounter" // Klucz, pod którym zapisujemy licznik
5.
6.     // Zmienna przechowująca stan licznika
7.     private var clickCount = 0
8.
9.     // Zapisywanie stanu licznika przy zmianie orientacji lub zamknięciu aplikacji
10.    override fun onSaveInstanceState(outState: Bundle) {
11.        super.onSaveInstanceState(outState)
12.        outState.putInt(COUNTER_KEY, clickCount)
13.        logAndAppend("onSaveInstanceState called")

```

```

14.     }
15.
16.     // Przywracanie stanu licznika po zmianie orientacji lub ponownym otwarciu aplikacji
17.     override fun onRestoreInstanceState(savedInstanceState: Bundle) {
18.         super.onRestoreInstanceState(savedInstanceState)
19.         clickCount = savedInstanceState.getInt(COUNTER_KEY, 0)
20.         logAndAppend("onRestoreInstanceState called")
21.     }
22.
23.     override fun onCreate(savedInstanceState: Bundle?) {
24.         super.onCreate(savedInstanceState)
25.         logAndAppend("onCreate called")
26.         setContentView(R.layout.activity_main)
27.         // Przywrócenie stanu licznika po przywróceniu aktywności
28.         if (savedInstanceState != null) {
29.             clickCount = savedInstanceState.getInt(COUNTER_KEY, 0)
30.         }
31.         else{ // Przywrócenie licznika z zapisanych preferencji po zamknięciu aplikacji
32.             val sharedPreferences = getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
33.             clickCount = sharedPreferences.getInt(COUNTER_KEY, 0) // Domyślnie 0, jeśli brak danych
34.             logAndAppend("sharedPreferences readed")
35.         }
36.
37.         val textViewCounter: TextView = findViewById(R.id.textViewCounter)
38.         val buttonClick: Button = findViewById(R.id.buttonClick)
39.
40.         // Ustawienie początkowej wartości licznika
41.         textViewCounter.text = clickCount.toString()
42.
43.         // Obsługa kliknięcia przycisku
44.         buttonClick.setOnClickListener {
45.             clickCount++ // Zwiększ liczbę kliknięć
46.             textViewCounter.text = clickCount.toString() // Zaktualizuj widok
47.         }
48.     }
49.
50.     override fun onStart() {
51.         super.onStart()
52.         logAndAppend("onStart called")
53.     }
54.
55.     override fun onResume() {
56.         super.onResume()
57.         logAndAppend("onResume called")
58.     }
59.
60.     override fun onPause() {
61.         super.onPause()
62.         logAndAppend("onPause called")
63.     }
64.
65.     override fun onStop() {
66.         super.onStop()
67.         logAndAppend("onStop called")
68.     }
69.
70.     override fun onRestart() {
71.         super.onRestart()
72.         logAndAppend("onRestart called")
73.     }
74.
75.     override fun onDestroy() {
76.         super.onDestroy()
77.         logAndAppend("onDestroy called")
78.
79.         val sharedPreferences = getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
80.         with(sharedPreferences.edit()) {
81.             putInt(COUNTER_KEY, clickCount) // Zapisz stan licznika
82.             apply()
83.         }
84.     }
85.
86.     private fun logAndAppend(message: String) {
87.         Log.d(TAG, message)
88.     }
89. }
90.

```

Aplikacja wykorzystuje zarówno stan zapisany w preferencjach aplikacji zapisywanych w pamięci urządzenia mobilnego co pozwala persystować dane w przypadku wyłączenia aplikacji i ładować je przy jej ponownym uruchomieniu, jak i korzysta ze stanu instancji która przechowuje informacje w ramach raz uruchomionej aplikacji między jej stanami np. w przypadku zminimalizowania i wybudzenia odzyskamy zachowane w ten sposób informacje.

```
2025-03-15 20:12:27.362 15576-15576 MainActivity com.example.pumlab2 D onPause called
2025-03-15 20:12:27.371 15576-15576 MainActivity com.example.pumlab2 D onStop called
2025-03-15 20:12:27.372 15576-15576 MainActivity com.example.pumlab2 D onSaveInstanceState called
2025-03-15 20:12:27.376 15576-15576 MainActivity com.example.pumlab2 D onDestroy called
2025-03-15 20:12:27.378 15576-15576 WindowOnBackDispatcher com.example.pumlab2 W sendCancelIfRunning: isInProgress
2025-03-15 20:12:27.531 15576-15576 MainActivity com.example.pumlab2 D onCreate called
2025-03-15 20:12:27.537 15576-15576 MainActivity com.example.pumlab2 D onStart called
2025-03-15 20:12:27.540 15576-15576 MainActivity com.example.pumlab2 D onRestoreInstanceState called
2025-03-15 20:12:27.544 15576-15576 MainActivity com.example.pumlab2 D onResume called
```

## activity\_main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent">
5.
6.     <!-- TextView do wyświetlania liczby kliknięć -->
7.     <TextView
8.         android:id="@+id/textViewCounter"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:text="0"
12.        android:textSize="30sp"
13.        android:layout_centerHorizontal="true"
14.        android:layout_marginTop="100dp"/>
15.
16.     <!-- Button do kliknięcia i zwiększania licznika -->
17.     <Button
18.         android:id="@+id/buttonClick"
19.         android:layout_width="wrap_content"
20.         android:layout_height="wrap_content"
21.         android:text="Kliknij"
22.         android:layout_below="@id/textViewCounter"
23.         android:layout_centerHorizontal="true"
24.         android:layout_marginTop="30dp"/>
25. </RelativeLayout>
26.
```

Interfejs użytkownika składa się z dwóch głównych elementów:

- **Button** – przycisk, który po naciśnięciu inkrementuje licznik.
- **TextView** – pole tekstowe, które wyświetla wartość licznika.

25

Kliknij

## Funkcje

- **Obsługa przycisku** – po naciśnięciu przycisku aplikacja inkrementuje wartość przechowywaną w pamięci i aktualizuje stan etykiety wyświetlającej aktualną wartość elemencie TextView.
- **Zapamiętywanie stanu** – w odpowiednich sytuacjach aplikacja zapisuje aktualną wartość licznika poprzez zapisanie stanu instancji w przypadku zmiany stanu aktywności, oraz w preferencjach aplikacji w przypadku jej zamknięcia.
- **Przywracanie stanu** – aplikacja przywraca stan licznika zapisany w instancji w przypadku przywrócenia aktywności, a w przypadku uruchomienia aplikacji z danych zapisanych na urządzeniu mobilnym.

## Testowanie

Aplikacja była testowana na wbudowanym emulatorze Android Studio. Testowano:

- Inkrementację stanu licznika w pamięci aplikacji.
- Zapisywanie stanu licznika w instancji, oraz pamięci urządzenia .
- Odczytywanie stanu licznika z pamięci urządzenia oraz stanu instancji.

## Wyniki

Aplikacja działa zgodnie z założeniami. Licznik inkrementuje się przy kliknięciu na przycisk, a jego stan jest zachowywany między stanami aplikacji, jak i w przypadku jej zamknięcia.

## Podsumowanie

Ćwiczenie nie było trudne dzięki dobrze opisanemu przykładowi użyciu stanów aplikacji, oraz przeciążania odpowiednich zdarzeń jak onCreate, OnDestroy w instrukcji do laboratorium, co pozwoliło łatwo wykonać zadanie stworzenia licznika przechowującego swój stan niezależnie od okoliczności jak zamknięcie, oraz minimalizacja aplikacji.

## Trudności i błędy

Zadanie nie sprawiło żadnych trudności.

## Źródła i odniesienia

Nie korzystano ze źródeł innych niż instrukcja.

## Dodatkowe materiały

Nie korzystano z dodatkowych materiałów.