


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	7	Data wykonania	03.05.2025	Grupa	1
Ocena		Data oddania	03.05.2025	Imię Nazwisko	Cezary Tytko
Nazwa ćwiczenia	Interfejs gry, tworzenie i obsługa interfejsu, intencje				

## 2. Cel ćwiczenia

Celem tego ćwiczenia laboratoryjnego jest nauczenie studentów podstawowych technik programowania aplikacji mobilnych na platformę Android przy użyciu języka Java. Studenci będą projektować i rozwijać interaktywną grę opartą na GridLayout, w której będą używać przycisków do interakcji z użytkownikiem. W projekcie zastosowano kilka ważnych elementów programowania. Po pierwsze, studenci zapoznają się z tworzeniem interfejsu użytkownika przy użyciu layoutów takich jak LinearLayout i GridLayout. Następnie nauczą się manipulować właściwościami widoków, takimi jak ustawianie tła, marginesów oraz kolorów, a także dodawanie tagów do widoków. Ważnym aspektem będzie obsługa zdarzeń dotykowych i przycisków, poprzez implementację `setOnClickListener` i dodawanie animacji do przycisków. Studenci dowiedzą się również, jak zarządzać cyklem życia aktywności, korzystając z metod `onCreate` oraz nowoczesnego podejścia do obsługi przycisku "Wstecz" przy pomocy `OnBackPressedCallback`. Kolejnym elementem jest losowanie i zarządzanie danymi gry, w tym tworzenie losowych kolorów, parowanie ich i aktualizacja stanu gry.

## 3. Opis projektu

Na podstawie przedstawionego poniżej szkieletu aplikacji implementującego pętle gry należy uzupełnić rozgrywkę inspirowaną klasyczną grą Memory. Jest to gra, gdzie wymieszane kartoniki należy ułożyć obrazkami skierowanymi w dół. Następnie losujemy dwa z nich – jeśli są takie same, zabieramy je jako zdobytą parę przez nas i możemy losować ponownie. Jeśli są różne – odkładamy je na miejsce, a ruch należy do przeciwnika. Wygrywa osoba, która zdobędzie najwięcej par. W wersji na urządzenia mobilne zadaniem gracza może być jak najszybsze odkrycie wszystkich par.

## 4. Implementacja

Kod został napisany w języku Kotlin w środowisku Android Studio.

Zapisywanie wyników w lokalnej bazie danych zostało zrealizowane za pomocą biblioteki Room.

MainActivity.kt:

```

1. package com.example.pumlab7
2.
3. import android.content.Intent
4. import android.os.Bundle
5. import android.os.CountDownTimer

```

```

6. import android.widget.Button
7. import android.widget.Toast
8. import androidx.activity.enableEdgeToEdge
9. import androidx.appcompat.app.AppCompatActivity
10. import androidx.core.view.ViewCompat
11. import androidx.core.view.WindowInsetsCompat
12. import com.example.pumlab7.databinding.ActivityMainBinding
13. import java.util.Date
14.
15. class MainActivity : AppCompatActivity() {
16.     private lateinit var binding: ActivityMainBinding
17.     private lateinit var memoryGame: MemoryGame
18.     private lateinit var timer: CountDownTimer
19.     private var startTime = 0L
20.     private var elapsedTime = 0L
21.     private var difficulty = 4 // liczba par (4x2)
22.
23.     override fun onCreate(savedInstanceState: Bundle?) {
24.         super.onCreate(savedInstanceState)
25.         binding = ActivityMainBinding.inflate(layoutInflater)
26.         setContentView(binding.root)
27.
28.         setupGame()
29.
30.         binding.resetButton.setOnClickListener {
31.             setupGame()
32.         }
33.         binding.highScoresButton.setOnClickListener {
34.             val intent = Intent(this, ScoresActivity::class.java)
35.             startActivity(intent)
36.         }
37.
38.         binding.easyButton.setOnClickListener {
39.             difficulty = 4 // 4 pary (8 kart)
40.             setupGame()
41.         }
42.
43.         binding.hardButton.setOnClickListener {
44.             difficulty = 8 // 8 par (16 kart)
45.             setupGame()
46.         }
47.     }
48.
49.     private fun setupGame() {
50.         memoryGame = MemoryGame(difficulty)
51.         startTime = System.currentTimeMillis()
52.         startTimer()
53.         displayBoard()
54.     }
55.
56.     private fun startTimer() {
57.         timer = object : CountDownTimer(Long.MAX_VALUE, 1000) {
58.             override fun onTick(millisUntilFinished: Long) {
59.                 elapsedTime = (System.currentTimeMillis() - startTime) / 1000
60.                 binding.timerText.text = "Czas: ${elapsedTime}s"
61.             }
62.
63.             override fun onFinish() {}
64.         }.start()
65.     }
66.
67.     private fun displayBoard() {
68.         binding.gridLayout.removeAllViews()
69.         val columns = 4
70.         binding.gridLayout.columnCount = columns
71.
72.         memoryGame.cards.forEachIndexed { index, card ->
73.             val button = Button(this).apply {
74.                 text = ""
75.                 setOnClickListener {
76.                     memoryGame.flipCard(index)
77.                     updateButtons()
78.                     if (memoryGame.hasWon()) {
79.                         timer.cancel()
80.                         saveScore()
81.                         Toast.makeText(this@MainActivity, "Wygrana! Czas: $elapsedTime s",
82. Toast.LENGTH_LONG).show()

```

```

83.         }
84.     }
85. }
86.     binding.gridLayout.addView(button)
87. }
88.     updateButtons()
89. }
90.
91.     private fun updateButtons() {
92.         for (i in 0 until binding.gridLayout.childCount) {
93.             val button = binding.gridLayout.getChildAt(i) as Button
94.             val card = memoryGame.cards[i]
95.             button.text = if (card.isFaceUp || card.isMatched) card.value.toString() else ""
96.             button.isEnabled = !card.isMatched
97.         }
98.     }
99.
100.     private fun saveScore() {
101.         val score = Score(0, elapsedTime, Date(), difficulty)
102.         Thread {
103.             ScoreDatabase.getDatabase(this).scoreDao().insert(score)
104.         }.start()
105.     }
106. }
107.
108.

```

## MemoryGame.kt:

```

1. package com.example.pumlab7
2.
3. data class Card(val id: Int, val value: Int, var isFaceUp: Boolean = false, var isMatched: Boolean = false)
4.
5. class MemoryGame(private val pairs: Int) {
6.     val cards: List<Card>
7.     private var indexOfSingleSelectedCard: Int? = null
8.
9.     init {
10.         val values = (1..pairs).flatMap { listOf(it, it) }.shuffled()
11.         cards = values.mapIndexed { i, v -> Card(i, v) }
12.     }
13.
14.     fun flipCard(position: Int) {
15.         val card = cards[position]
16.         if (card.isMatched || card.isFaceUp) return
17.
18.         if (indexOfSingleSelectedCard == null) {
19.             restoreCards()
20.             card.isFaceUp = true
21.             indexOfSingleSelectedCard = position
22.         } else {
23.             val matchedCard = cards[indexOfSingleSelectedCard!!]
24.             if (matchedCard.value == card.value) {
25.                 matchedCard.isMatched = true
26.                 card.isMatched = true
27.             }
28.             card.isFaceUp = true
29.             indexOfSingleSelectedCard = null
30.         }
31.     }
32.
33.     fun restoreCards() {
34.         cards.filter { !it.isMatched }.forEach { it.isFaceUp = false }
35.     }
36.
37.     fun hasWon(): Boolean = cards.all { it.isMatched }
38. }
39.
40.

```

## Implementacja bazy danych i zapisywania wyników: ScoreDatabase.kt:

```
1. package com.example.pumlab7
2.
3. import android.content.Context
4. import androidx.lifecycle.LiveData
5. import androidx.room.Dao
6. import androidx.room.Database
7. import androidx.room.Entity
8. import androidx.room.Insert
9. import androidx.room.PrimaryKey
10. import androidx.room.Query
11. import androidx.room.Room
12. import androidx.room.RoomDatabase
13. import androidx.room.TypeConverter
14. import androidx.room.TypeConverters
15. import java.util.Date
16.
17. @Entity
18. data class Score(
19.     @PrimaryKey(autoGenerate = true) val id: Int = 0,
20.     val time: Long,
21.     val date: Date,
22.     val difficulty: Int
23. )
24.
25. @Dao
26. interface ScoreDao {
27.     @Insert
28.     fun insert(score: Score)
29.     @Query("SELECT * FROM Score ORDER BY difficulty Desc, time ASC LIMIT 10")
30.     fun getBestScores(): LiveData<List<Score>>
31. }
32.
33. @Database(entities = [Score::class], version = 2)
34. @TypeConverters(Converters::class)
35. abstract class ScoreDatabase : RoomDatabase() {
36.     abstract fun scoreDao(): ScoreDao
37.
38.     companion object {
39.         private var INSTANCE: ScoreDatabase? = null
40.
41.         fun getDatabase(context: Context): ScoreDatabase {
42.             return INSTANCE ?: synchronized(this) {
43.                 val instance = Room.databaseBuilder(
44.                     context.applicationContext,
45.                     ScoreDatabase::class.java,
46.                     "score_database"
47.                 )
48.                     .fallbackToDestructiveMigration()
49.                     .build()
50.                 INSTANCE = instance
51.                 instance
52.             }
53.         }
54.     }
55. }
56.
57. class Converters {
58.     @TypeConverter
59.     fun fromTimestamp(value: Long?): Date? = value?.let { Date(it) }
60.     @TypeConverter
61.     fun dateToTimestamp(date: Date?): Long? = date?.time
62. }
63.
```

## ScoresActivity.kt:

```
1. package com.example.pumlab7
2.
3. import android.os.Bundle
4. import androidx.appcompat.app.AppCompatActivity
5. import androidx.recyclerview.widget.LinearLayoutManager
```

```

6. import androidx.recyclerview.widget.RecyclerView
7. import com.example.pumlab7.databinding.ActivityScoresBinding
8.
9. class ScoresActivity : AppCompatActivity() {
10.     private lateinit var binding: ActivityScoresBinding
11.     private lateinit var recyclerView: RecyclerView
12.     private lateinit var adapter: ScoreAdapter
13.
14.     override fun onCreate(savedInstanceState: Bundle?) {
15.         super.onCreate(savedInstanceState)
16.         binding = ActivityScoresBinding.inflate(layoutInflater)
17.         setContentView(binding.root)
18.
19.         recyclerView = findViewById(R.id.recyclerView)
20.         recyclerView.layoutManager = LinearLayoutManager(this)
21.
22.         val db = ScoreDatabase.getDatabase(this)
23.         db.scoreDao().getBestScores().observe(this) { scores ->
24.             adapter = ScoreAdapter(scores)
25.             recyclerView.adapter = adapter
26.         }
27.
28.         binding.backButton.setOnClickListener {
29.             finish() // zamyka aktywność i wraca do MainActivity
30.         }
31.
32.     }
33. }
34.
35.

```

### ScoreAdapter.kt:

```

1. package com.example.pumlab7
2.
3. import android.view.LayoutInflater
4. import android.view.View
5. import android.view.ViewGroup
6. import android.widget.TextView
7. import androidx.recyclerview.widget.RecyclerView
8. import java.text.SimpleDateFormat
9. import java.util.Locale
10.
11. class ScoreAdapter(private val scores: List<Score>) :
12.     RecyclerView.Adapter<ScoreAdapter.ScoreViewHolder>() {
13.
14.     class ScoreViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
15.         val scoreText: TextView = itemView.findViewById(R.id.score)
16.         val difficulty: TextView = itemView.findViewById(R.id.difficulty)
17.         val date: TextView = itemView.findViewById(R.id.date)
18.     }
19.
20.     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ScoreViewHolder {
21.         val view = LayoutInflater.from(parent.context)
22.             .inflate(R.layout.item_score, parent, false)
23.         return ScoreViewHolder(view)
24.     }
25.
26.     override fun onBindViewHolder(holder: ScoreViewHolder, position: Int) {
27.         val score = scores[position]
28.         val sdf = SimpleDateFormat("yyyy-MM-dd HH:mm", Locale.getDefault())
29.         holder.date.text = sdf.format(score.date)
30.         holder.difficulty.text = score.difficulty.toString()
31.         holder.scoreText.text = score.time.toString()
32.     }
33.
34.     override fun getItemCount() = scores.size
35. }
36.
37.

```

### Activitymain.xml:

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:id="@+id/rootLayout"

```

```

4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:orientation="vertical"
7.     android:padding="16dp">
8.
9.     <TextView
10.         android:id="@+id/timerText"
11.         android:layout_width="wrap_content"
12.         android:layout_height="wrap_content"
13.         android:text="Czas: 0s" />
14.
15.     <GridLayout
16.         android:id="@+id/gridLayout"
17.         android:layout_width="match_parent"
18.         android:layout_height="0dp"
19.         android:layout_weight="1"
20.         android:alignmentMode="alignMargins"
21.         android:columnCount="4"
22.         android:rowCount="4"
23.         android:useDefaultMargins="true" />
24.
25.     <Button
26.         android:id="@+id/highScoresButton"
27.         android:layout_width="wrap_content"
28.         android:layout_height="wrap_content"
29.         android:text="Wyniki" />
30.
31.     <Button
32.         android:id="@+id/easyButton"
33.         android:layout_width="wrap_content"
34.         android:layout_height="wrap_content"
35.         android:text="Łatwy" />
36.
37.     <Button
38.         android:id="@+id/hardButton"
39.         android:layout_width="wrap_content"
40.         android:layout_height="wrap_content"
41.         android:text="Trudny" />
42.
43.     <Button
44.         android:id="@+id/resetButton"
45.         android:layout_width="match_parent"
46.         android:layout_height="wrap_content"
47.         android:text="Resetuj grę" />
48. </LinearLayout>
49.
50.

```

#### Activityscores.xml:

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical">
6.
7.     <Button
8.         android:id="@+id/backButton"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:text="Powrót do gry"
12.        android:layout_marginBottom="8dp" />
13.
14.     <LinearLayout
15.         android:layout_width="match_parent"
16.         android:layout_height="wrap_content"
17.         android:orientation="horizontal"
18.         android:padding="8dp">
19.
20.         <TextView
21.             android:layout_width="0dp"
22.             android:layout_height="wrap_content"
23.             android:layout_weight="1"
24.             android:text="Data"
25.             android:gravity="center" />
26.
27.         <TextView
28.             android:layout_width="0dp"
29.             android:layout_height="wrap_content"

```

```

30.         android:layout_weight="1"
31.         android:text="Trudność"
32.         android:gravity="center" />
33.
34.     <TextView
35.         android:layout_width="0dp"
36.         android:layout_height="wrap_content"
37.         android:layout_weight="1"
38.         android:text="Czas"
39.         android:gravity="center"/>
40.
41. </LinearLayout>
42.
43. <androidx.recyclerview.widget.RecyclerView
44.     android:id="@+id/recyclerView"
45.     android:layout_width="match_parent"
46.     android:layout_height="0dp"
47.     android:layout_weight="1" />
48.
49. </LinearLayout>
50.
51.

```

#### Item\_score.xml:

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:layout_width="match_parent"
5.     android:layout_height="wrap_content"
6.     android:orientation="horizontal"
7.     android:padding="8dp">
8.
9.     <TextView
10.         android:id="@+id/date"
11.         android:layout_width="0dp"
12.         android:layout_weight="1"
13.         android:layout_height="wrap_content"
14.         android:gravity="center"/>
15.
16.     <TextView
17.         android:id="@+id/difficulty"
18.         android:layout_width="0dp"
19.         android:layout_weight="1"
20.         android:layout_height="wrap_content"
21.         android:gravity="center"/>
22.
23.     <TextView
24.         android:id="@+id/score"
25.         android:layout_width="0dp"
26.         android:layout_weight="1"
27.         android:layout_height="wrap_content"
28.         android:gravity="center"/>
29. </LinearLayout>
30.

```

## 5. Funkcje kluczowe

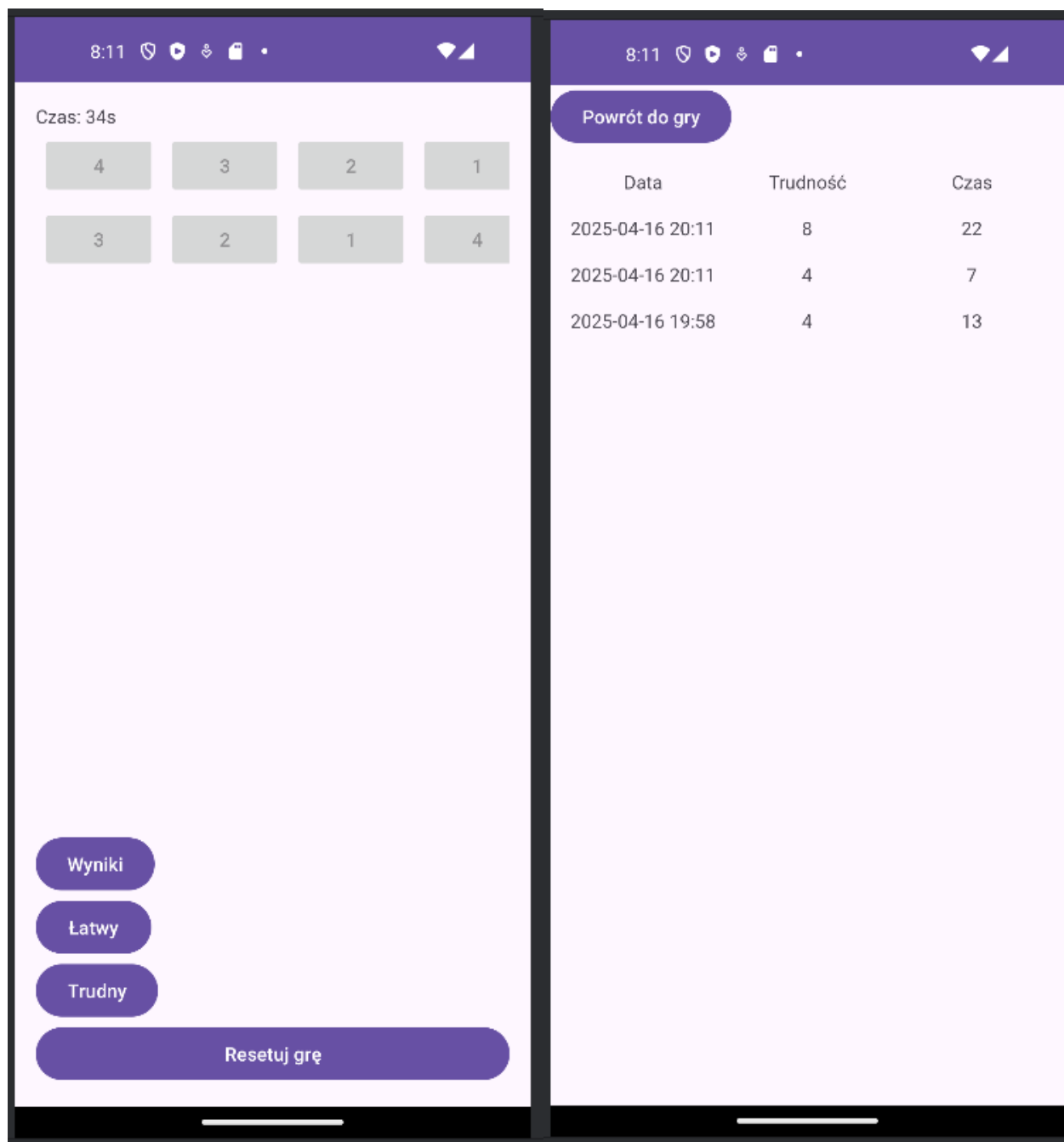
- Obsługa gry – wyświetlanie cyfr i łączenie w pary.
- Zapisywanie wyników w lokalnej bazie danych.
- Wyświetlanie wyników – data, poziom trudności, czas.

## 6. Testowanie

Testowanie gry obejmowało:

- Sprawdzenie, czy liczby poprawnie łączone są w pary.
- Sprawdzenie obsługi zdarzenia wygranej.
- Sprawdzenie, czy wyniki zapisują się poprawnie w lokalnej bazie danych nawet po zamknięciu aplikacji.
- Obsługa różnych poziomów trudności.

## 7. Wyniki



## 8. Podsumowanie

Projekt realizuje klasyczną grę pamięciową opartą na łączeniu liczb w pary w aplikacji mobilnej. Implementacja pozwoliła na zdobycie doświadczenia w tworzeniu aplikacji na Androida, obsłudze interakcji użytkownika oraz zarządzaniu stanem gry. Ćwiczenie umożliwiło lepsze zrozumienie obsługi zdarzeń i logiki gry.

## 9. Trudności i błędy

- Nie wystąpiły żadne trudności ani błędy.

## 10. Źródła i odniesienia

- Nie korzystano ze źródeł i odniesień innych niż ta instrukcja.

## 11. Dodatkowe materiały

- Nie korzystano z dodatkowych materiałów.