# Szymon Walkusz Gr. 215 IC_b2

## LABORATORIUM nr 3 – Aplikacje mobilne (2 aktywności)

### Polecenie:

Wykonać aplikację mobilną wczytującą dane liczbowe oraz sortującą je zgodnie z wybranymi metodami sortowania.

Aplikacja ma być wykonana w technologii obiektowej i umożliwiać wybór metody sortowania. Należy zaimplementować

- dwie wybrane metody sortowania:
- bąbelkowe,
- przez wstawianie,
- przez scalanie,
- przez wybór,
- szybkie

Efekty działania aplikacji:

- podanie metody sortowania
- krótki opis metody sortowania
- początkowy ciąg
- posortowany ciąg
- krótkie sprawozdanie w wersji elektronicznej

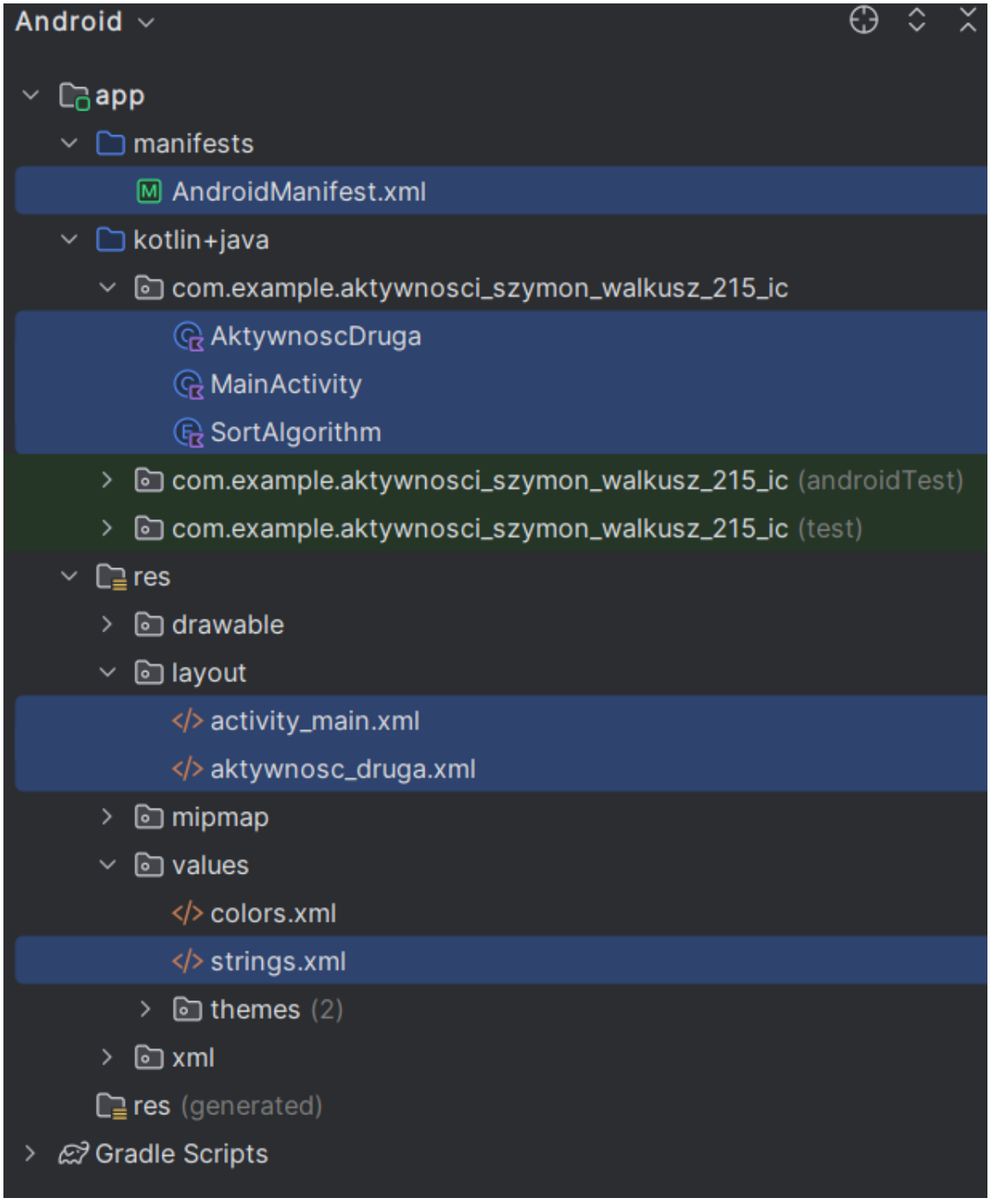
Należy wykorzystać poznane elementy Android Studio i języka Kotlin.

Aplikacja ma być wykonana w dwóch wersjach:

1) **Zastosowanie dwóch aktywności (activity)**
    a) **Główna aktywność ustala co mamy do wykonania i przekazuje parametry do aktywności podrzędnej**
    b) **Aktywność wywoływana wykonuje wszystkie działania i wizualizuje efekty.**
2) Zastosowanie jednej aktywności i dwóch fragmentów
    a) Pierwszy fragment ustala co trzeba wykonać
    b) Drugi fragment wykonuje zadanie i wizualizuje efekty

# Opis

Podczas tworzenia aplikacji korzystano z wielu plików Android Studio, które zostały wyszczególnione niebieskim kolorem na Rys 1.




***Rys. 1.*** *Struktura projektu*

***Tabela 1.*** *Opis zadań poszczególnych plików*

| Nazwa pliku | Wykonywane zadania |
|---|---|
| AndroidManifest.xml | Rejestracja i obsługa intencji i aktywności |
| AktywnoscDruga.kt | Plik z zawierający logikę dla drugiej aktywności. Pobiera dane z przekazanej mu aktywności, podejmuje odpowiednie działania na podstawie przekazanych danych, takich jak dobór odpowiedniego algorytmu sortowania, ustawienie nowego tekstu dla elementów graficznych. |
| MainActivity.kt | Plik z zawierający logikę dla pierwszej aktywności. Pobiera odpowiednie dane z elementów graficznych, tworzy intencję z danymi do przekazania i wysyła ją do drugiej aktywości. |
| SortAlgorithm.kt | Klasa typu wyliczeniowego zawierająca nazwy dostępnych algorytmów sortowania. |
| activity_main.xml | Plik określający elementy graficzne dla aktywności startowej. Znajdują się tu przyciski typu radio button, button, pole tekstowe przyjmujące liczby. |
| aktywnosc_druga.xml | Plik określający elementy graficzne dla drugiej aktywności. Zawiera ona elementy wyświetlające tekst z podsumowaniem wykonanych akcji i przycisk przekierowujący do pierwszej aktywności. |
| strings.xml | Plik ze stałymi, które zawierają frazy tekstu, dostępne do wielokrotnego wykorzystania w plikach określających wygląd graficzny. |

```
14  class MainActivity : AppCompatActivity() {
15
16      private lateinit var sortButton: Button
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          enableEdgeToEdge()
21          setContentView(R.layout.activity_main)
22          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, in
23              val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars()
24              v.setPadding(systemBars.left, systemBars.top, systemBars.right, system
25              insets ^setOnApplyWindowInsetsListener
26          }
27
28          sortButton = findViewById(R.id.sortedButton)
29          val editText = findViewById<EditText>(R.id.inputField)
30          val radioGroup = findViewById<RadioGroup>(R.id.sortMethodRadioGroup)
31
32  //          function to handle on a click
33          sortButton.setOnClickListener { it: View!
34              val inputText = editText.text.toString()
35              val selectedSortAlgorithm = when (radioGroup.checkedRadioButtonId) {
36                  R.id.sort1RadioButton -> SortAlgorithm.BUBBLE_SORT.toString()
37                  R.id.sort2RadioButton -> SortAlgorithm.SELECTION_SORT.toString()
38                  else -> SortAlgorithm.BUBBLE_SORT.toString()
39              }
40              intent = Intent( packageContext: this, AktywnoscDruga::class.java)
41              intent.putExtra( name: "algorithm", selectedSortAlgorithm)
42              intent.putExtra( name: "inputText", inputText)
43              startActivity(intent)
44          }
45      }
```

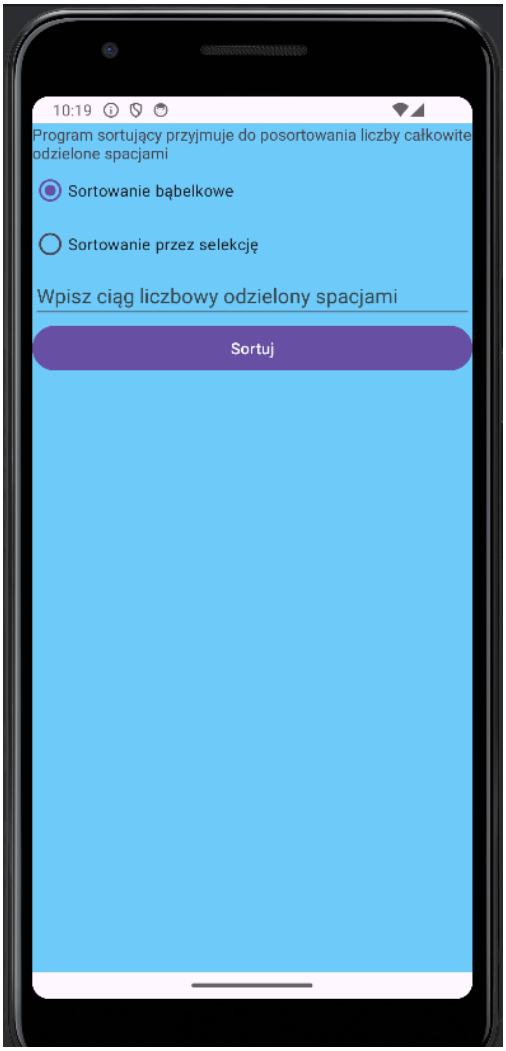
***Rys. 2.*** *Fragment kodu z pliku MainActivity.kt*

Po przyciśnięciu przycisku sortButton, następuje pobranie danych z części graficznej, a następnie utworzenie intencji i przekazanie parametrów (linijki 40-43).

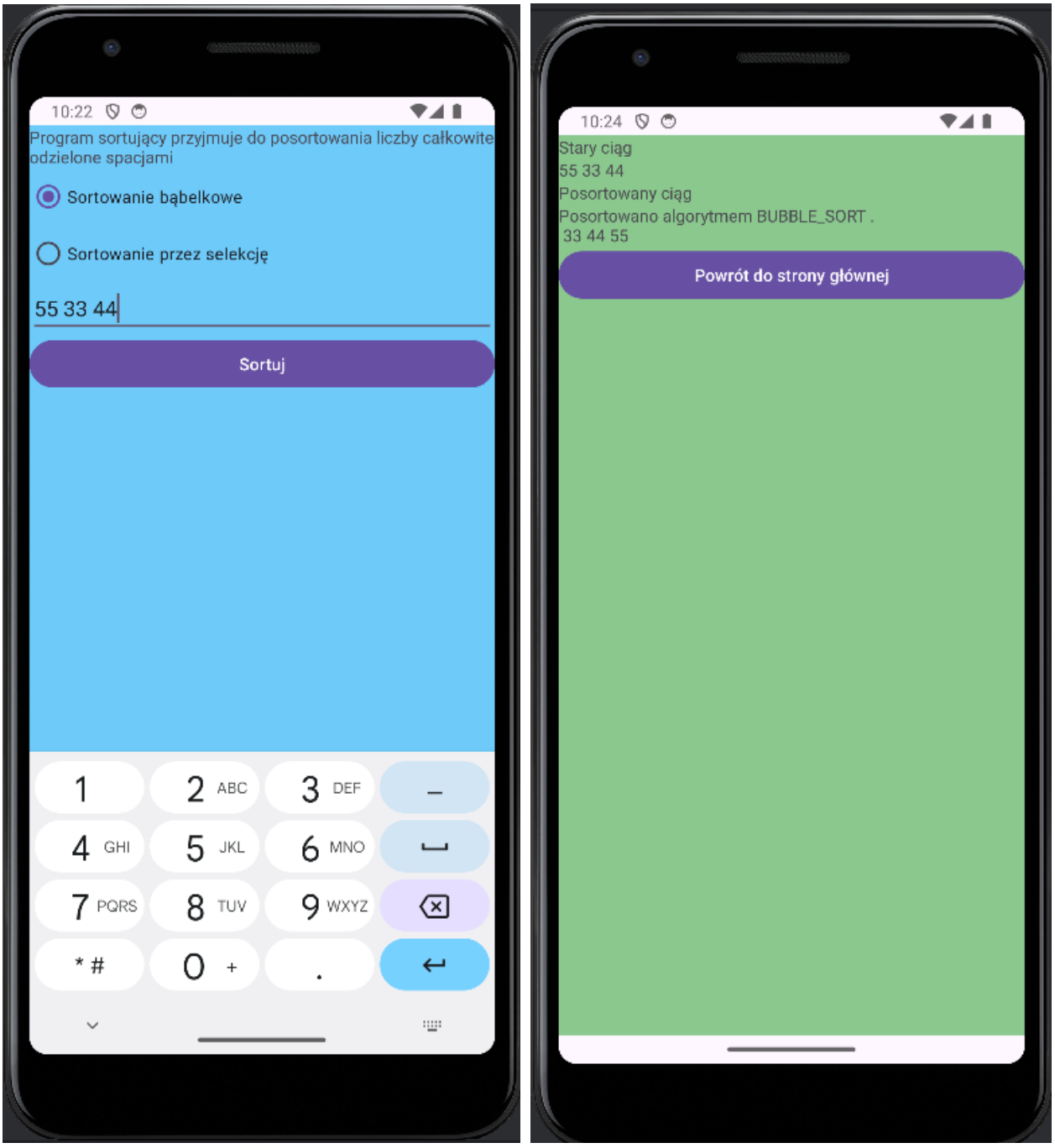
```
17      @SuppressLint("MissingInflatedId", "SetTextI18n")
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          enableEdgeToEdge()
21          setContentView(R.layout.aktywnosc_druga)
22          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
23              val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
24              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.botto
25              insets ^setOnApplyWindowInsetsListener
26          }
27
28          homePageBtn = findViewById(R.id.backToHomeButton)
29          val extras = intent.extras ?: return
30          val algorithm = extras.getString( key: "algorithm")
31          val inputText = extras.getString( key: "inputText")
32
33
34          val oldChain: TextView = findViewById(R.id.oldChain)
35          oldChain.text = inputText
36          val sortedChain: TextView = findViewById(R.id.sortedChain)
37
38          val sortedNumbers = when (algorithm) {
39              SortAlgorithm.BUBBLE_SORT.toString() -> inputText?.let { bubbleSort(it) }
40              SortAlgorithm.SELECTION_SORT.toString() -> inputText?.let { selectionSort(it) }
41              else -> "nie dziala: $algorithm"
42          }
43          sortedChain.text = "Posortowano algorytmem $algorithm .\n $sortedNumbers"
44
45          homePageBtn.setOnClickListener { it: View!
46              intent = Intent( packageContext: this, MainActivity::class.java)
47              startActivity(intent)
48          }
```

***Rys. 3.*** *Fragment kodu z pliku AktywnoscDruga.kt*

Linijki 29-31 przekazują dane z przekazanej intencji do zmiennych. Następnie następują akcje poszukiwania elementów, uruchomienie odpowiedniej metody sortowania, wypisanie wyników. Linie 45-47 są odpowiedzialne za przekierowanie użytkownika do widoku startowego.




***Rys. 4.*** *Widok ekranu użytkownika po uruchomieniu aplikacji*.




***Rys. 5 i 6.*** *- Widok działającej aplikacji po wprowadzeniu danych*

## Wnioski

Dzięki zastosowaniu intencji w projekcie, możliwe było przesyłanie danych pomiędzy aktywnościami oraz uruchamianie innych aktywności zawartych w tym samym projekcie.

Właściwości kolejnych plików opisano w poprzednich sprawozdaniach.