Obsługa widoku w Androidzie

Anna Gogolińska

Odnośnie programowania w Java

- Typy zmiennych są prawie takie same jak w C++. Różnice:
  - Napis to String (a nie string), boolean, a nie bool.
- Instrukcje warunkowe i pętle takie same jak w C++.
- Operatory (+=, -=, ++, -- itd.) takie same jak w C++.
- Wypisywanie na ekran:
  - System.out.println("napis " + zmienna + " napis"); - wypisanie i przejście do nowej linii.
  - System.out.println(); - przejście do nowej linii (linia odstępu).
  - System.out.print(zmienna); - wypisanie bez przejścia do nowej linii.
  - Zmienne i tekst w wypisywaniu łączy się + (jak w JavaScript).
  - Skrót **sout** (powoduje wstawienie System.out.println()).
- W niektórych miejscach trzeba użyć Integer lub Double zamiast int czy double.

Odnośnie programowania w Java

- Nieraz kiedy wpiszemy nazwę klasy (typu) będzie ona na czerwono.
  - Należy dodać plik z klasą do projektu – odpowiednik include. Tu nazywa się to **import**.
  - Po najechaniu myszką pojawi się małe okienko, gdzie trzeba kliknąć „import class”.
  - Można również kliknąć Enter gdy piszemy nazwę klasy.
- Wiele elementów ma listę możliwych metod do użycia. Wystarczy napisać „nazwa.” i po „.” wyświetli się lista metod.
  - Dla napisów (String) mamy na przykład:
    - napis.contains(napis2) – sprawdza czy napis zawiera napis2.
    - napis.equals(napis2) – sprawdza czy napis się równy napis2 ("==" nie zadziała).

Losowanie

- Tworzenie obiektu do losowania:
  Random random = new Random();
- Losowanie liczby całkowitej od n do m (łącznie z n, ale bez m):
  int liczba = random.nextInt(n, m);
- Losowanie liczby zmiennoprzecinkowej od n do m (jak wyżej):
  double liczba2 = random.nextDouble(n, m);

Odwołanie się do Elementu (Widoku)

- Aby użyć elementu (widoku) w kodzie Java, musisz najpierw uzyskać do niego referencję.
  - Odbywa się to za pomocą metody **findViewById()** i przekazania ID elementu, które zostało mu nadane w pliku XML.
  - ID w xml ustawiane są automatycznie, ale można je zmienić. Są widoczne w pliku xml jako wartość dla: **android:id**:
    - android:id="@+id/**button**" – to co znajduje się po „/” to id.
  - W kodzie java należy id z pliku xml poprzedzić **R.id**:
    - **myTextView = findViewById(R.id.button);**

Pobranie i ustawienie wartości

- Jak mamy już odwołanie do elementu można pobrać jego wartość lub ją ustawić:
  - **TextView i EditText:**
    - Pobranie wartości i przypisanie jej do zmiennej:
      String napisz = pole.**getText().toString();**
    - Ustawienie wartości:
      pole.**setText**("aaa" + zmienna + " " + i);
  - **Suwak (SeekBar)**
    - Pobranie wartości i przypisanie do zmiennej:
      int aktualnaWartosc = mySeekBar.**getProgress();**
    - Ustawienie wartości (nowa wartość to liczba int):
      mySeekBar.**setProgress**(nowaWartosc);

Pobranie i ustawienie wartości

- **RadioButton i CheckBox:**
  - Pobranie i ustawienie napisów: **getText()** oraz **setText(**napis**)**.
  - Sprawdzenie zaznaczenia: isChecked():
    boolean isChecked = myCheckBox.**isChecked();**
  - Ustawienie zaznaczenia: **setChecked(***arg***)** gdzie *arg* to *true* albo *false*.
    checkBox.**setChecked(true);**
- **ImageView:**
  - Ustawienie nowej wartości obrazka (nowy plik – plik musi być w drawable):
    image.**setImageResource(**R.drawable.new_image);

Zmiana wyglądu z poziomu kodu

- Mając referencję do elementów można zmienić wygląd tego elementu:
  - Zmiana koloru tła:
    pole.**setBackgroundColor**(Color.GREEN);
    pole.setBackgroundColor(Color.parseColor("#FF6600"));
  - Zmiana koloru napisu:
    edit.**setTextColor**(Color.*BLUE*);
  - Zmiana wielkości napisu:
    text.**setTextSize**(30);
    text.setTextSize(TypedValue.COMPLEX_UNIT_SP, 30);

Reakcja na naciśnięcie przycisku

- Należy zdefiniować metodę w kodzie, która będzie public, będzie zwracać void i będzie miała argument View view.
  - Argument ten to będzie element, który naciśnięto.
  - **public void onWyslij(View view)** {
    System.out.println("Nacisnieto przycisk");
    }
- Dla przycisku (czy innego elementu) należy ustawić że ta metoda ma być reakcją na naciśnięcie, czyli **android:onClick** w pliku xml (lub w edytorze):

```
<Button
   android:id="@+id/buttonId"
   android:onClick="obsluzKliknieciePrzycisku"  />
```

Reakcja na przesuwanie suwaka

- Aby reagować na interakcję użytkownika z suwakiem SeekBar, musisz zdefiniować **seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener**….. Wymaga to definicji **trzech metod** (zostaną one wygenerowane, część może zostać pusta).

Metoda	Kiedy jest wywoływana?	Najważniejszy argument	Co się tu robi?
onProgressChanged	W trakcie każdego ruchu suwaka.	progress (aktualna wartość)	Aktualizowanie etykiet, przeliczanie parametrów.
onStartTrackingTouch	W momencie dotknięcia suwaka.	-	Przygotowanie interfejsu do zmiany.
onStopTrackingTouch	W momencie puszczenia suwaka.	-	Zapisanie ostatecznej wartości, wywołanie głównej akcji.

Reakcja na przesuwanie suwaka (w onCreate())

```
SeekBar mySeekBar = findViewById(R.id.seekBarId);
TextView infoTextView = findViewById(R.id.infoTextViewId);
mySeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
@Override
   public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
      //Wyświetlanie wartości w trakcie przesuwania
      String komunikat = "Nowa wartość: " + progress";
      infoTextView.setText(komunikat);
   }
@Override
   public void onStartTrackingTouch(SeekBar seekBar) {
      // Możesz tutaj np. zapisać początkową wartość
      System.out.println("Zaczęto przesuwanie.");
   }
@Override
   public void onStopTrackingTouch(SeekBar seekBar) {

   }
});
```