Лабораторна робота № 10

Тема: Реалізація обробки подій

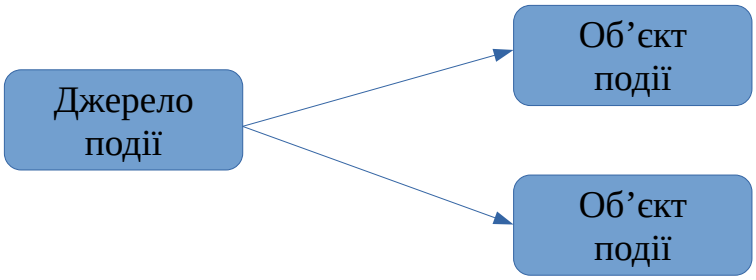
Мета: Вивчення принципів реалізації обробки подій на основі бібліотек Java

Теоретична частина

Проекти додатків з графічним інтерфейсом, як правило, потребує реалізації певної моделі обробки подій. Для створення інтерфейсу, який взаємодіє з користувачем, необхідно розмістити певні компоненти та забезпечити їх відповідною моделлю обробки подій. Останнє забезпечується за рахунок використання класів та інтерфейсів бібліотек Java. Зазвичай модель обробки подій асоціює кожен компонент з його спостерігачем таким чином, щоб у разі події про це повідомлялися всі його спостерігачі. Спостерігач є об'єктом, що очікує визначену подію.

Розглянемо принципи реалізації обробки подій на основі компонентів Swing. При використанні компонентів Swing спостерігачами є слухачі подій, що реалізовують порожній інтерфейс java.util.EventListener, а також інтерфейс слухача підкласу, який повинен мати хоча б один метод. Події мають створювати підклас класу java.util.EventObject.

Для прикладу, розглянемо випадок вибору кнопки JButton. За цю дію відповідає подія ActionEvent, що генерується при виборі користувачем кнопки JButton. Якщо певний об'єкт потребує інформування про виникнення цієї події, то його необхідно зареєструвати для кнопки JButton.



Для події ActionEvent реєстрація проводиться за формою ActionListener. Імена у ній пов'язано наступним чином: кожній події виду **ABC**Event асоціюється слух **ABC**Listener, де замість ABC буде конкретний тип події.

Реєстрація відбувається під час виклику методу addActionListener у об’єкту класу JButton. Кожен зареєстрований реалізатор отримує повідомлення шляхом передавання (при генерації події ActionEvent) реалізації інтерфейсу слухача методом addActionListener. В однієї події може бути кілька спостерігачів.

Процес представляється наступними пунктами:

- Визначити для заданої події клас, який реалізує пов'язаний із нею інтерфейс. До існуючого визначення класу додається "implements ABCListener" або створюється новий клас, що реалізує цей інтерфейс.

- Крім додавання до визначення класу "implements ABCListener", необхідно реалізувати кожен метод даного інтерфейсу. Деякі слухачі, наприклад, ActionListener, мають один метод, в інших, наприклад, WindowListener, їх декілька. Код можна компілювати тоді, коли визначено усі методи слухачів.
- Після визначення реалізації інтерфейсу необхідно створити екземпляр реалізації та асоціювати його з компонентом. Тільки після цього спостерігач (слухач) зможе отримувати сповіщення про виникнення відповідних подій.

Приклад 1

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

class MyActionListener implements ActionListener {
  @Override
  public void actionPerformed(ActionEvent e) {
    System.out.println("Thank you.");
  }
}

public class SelectMe extends JFrame{
    public SelectMe() {
    super("Hello");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    JButton button = new JButton("Pick Me");
    ActionListener listener = new MyActionListener();
    button.addActionListener(listener);
    getContentPane().add(button, BorderLayout.CENTER);
    setSize(200, 200);
  }

    public static void main(String[] args) {
        JFrame frame = new SelectMe();
        frame.setVisible(true);
    }
}
```

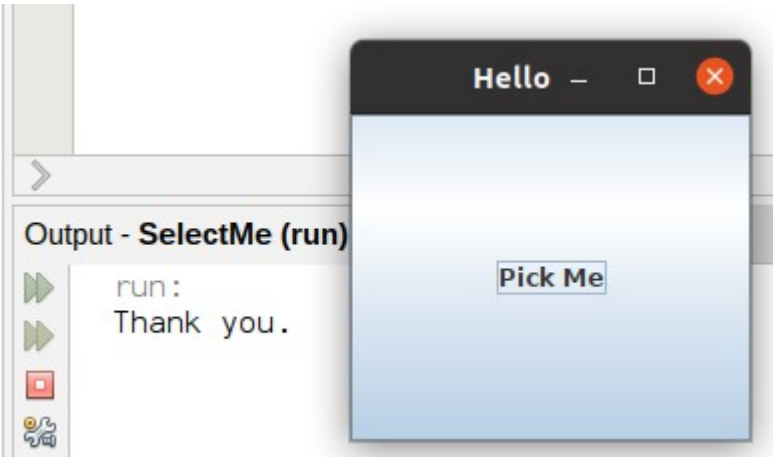


Рис.1 Результат роботи наведеної програми

Ініціація компонентом події додається методом з префіксом add. Наприклад, у наведеному прикладі компонентом типу JButton ініціюється подія addActionListener(listener), де listener — об’єкт класу MyActionListener.

Для відключення відправлення повідомлення про виникнення події використовується метод з префіксом remove, наприклад, removeActionListener.

Базис кода обробки подій для компонента JButton визначено у його суперкласі AbstractButton. На рівні AbstractButton слухачами, що асоціюються з класом JButton, крім ActionListener також можуть бути ChangeListener та ItemListener. ActionListener використовується для повідомлення, що компонент обрано, ChangeListener вказує на зміни властивостей кнопки. Властивості компоненту описують його стан. Властивостями класу JButton є текстова мітка, фоновий та основні кольори, а також шрифт. При зміні хоча б однієї властивості інформуються усі зареєстровані слухачі ChangeListener. Інформація про назву властивості, що змінилась, у події не передається. Слухач PropertyChangeListener виконує ті самі функції, але вказує на властивість, що змінилась.

Додамо до програми з прикладу 1 слухача ChangeListener. Тепер будь яка зміна вигляду кнопки буде ініціювати відправлення повідомлення. Таких подій може бути множина: зміна фонового кольору або рамки компоненту при наведені чи натискані на кнопку, тощо. Тому слухач ChangeListener неодноразово отримує повідомлення.

Приклад 2

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

class MyActionListener implements ActionListener {
  @Override
  public void actionPerformed(ActionEvent e) {
    System.out.println("Thank you.");
  }
}

class MyChangeListener implements ChangeListener {
  public void stateChanged(ChangeEvent e) {
    System.out.println("You're welcome.");
  }
}

public class SelectMe extends JFrame{
    public SelectMe() {
    super("Hello");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    JButton button = new JButton("Pick Me");
    ActionListener listener = new MyActionListener();
    button.addActionListener(listener);
```

```
        ChangeListener cListener = new MyChangeListener();
        button.addChangeListener(cListener);
        getContentPane().add(button, BorderLayout.CENTER);
        setSize(200, 200);
      }

        public static void main(String[] args) {
            JFrame frame = new SelectMe();
            frame.setVisible(true);
        }
    }
```

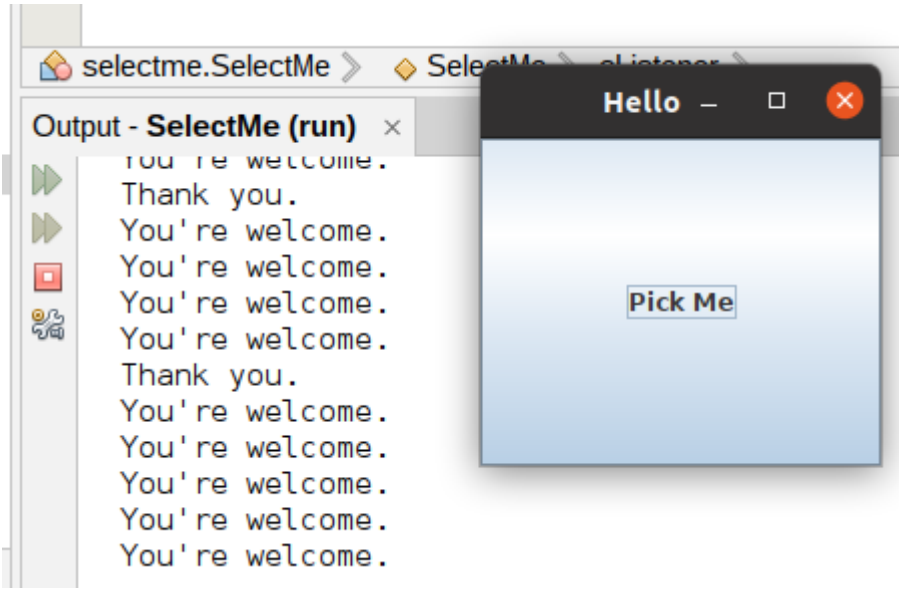


Рис.2 Будь-яка зміна властивостей викликає події, що оброблюються

У мобільних додатках для реалізації слухачів подій, що генеруються такими компонентами інтерфейсу, як кнопки використовуються подібні, розглянутим вище, засоби, але реалізовані у відповідних класах інших бібліотек, наприклад, у Framework Android. Для реалізації кнопки у додатку Android виконується імпорт відповідної бібліотеки

import android.widget.Button;

та створюється змінна посилання на об’єктів

private Button **mButton**;

Для мобільних додатків Android графічний інтерфейс створюється за допомогою xml файлів макета (layout), в яких необхідні віджети організовано у певну ієрархію та з необхідними параметрами. Змінна посилання зв’язується з об’єктом відповідного віджета, наприклад, для кнопки таке зв’язування буде виглядати подібним чином:

mButton = (Button)findViewById(R.id.***buttonFirst***);

Наступним кроком може бути утворений слухач відповідної події, наприклад, натискання кнопки:

```
mButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        System.out.println("Button is OnClick");
    }
});
```

Очевидно, що і реалізація компонент графічного інтерфейсу, і реалізація слухачів подій цих компонент для додатків для десктопів і мобільних додатків суттєво відрізняються, що обумовлюється використанням різних бібліотек. Тому, кросплатформеність для компонент графічного інтерфейсу може досягатись тільки на рівні алгоритмів обробки подій, що генеруються цими компонентами. Програмна реалізація компонент графічного інтерфейсу, генераторів та обробників подій суттєво залежить від фреймворків, що використовуються для розробки додатків. Для реалізації кросплатформності додатків з графічним інтерфейсом необхідно враховувати особливості програмної архітектури фреймворків тих платформ, для яких вони реалізовуються.

Завдання

1. Вивчить теоретичний матеріал лабораторної роботи та реалізуйте наведені приклади.
2. Подібно до прикладів 1 та 2 створіть подібний інтерфейс для Android додатка.
3. Для декстоп додатку із першого прикладу та Android додатка змініть реакцію на натискання кнопки — додатки повинні виконати розрахунок будь-якої математичної функції (однакової для обох додатків) за одними і тими параметрами.
4. У чому схожість і відмінність коду цих додатків?
5. Підготуйте звіт, у якому наведіть створений код програм та скриншоти їх роботи.