Programowanie obiektowe

Wykład 11: Obsługa zdarzeń

dr inż. Marcin Luckner
mluckner@mini.pw.edu.pl

Wydział Matematyki i Nauk Informacyjnych

Wersja 1.3
4 marca 2021

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny

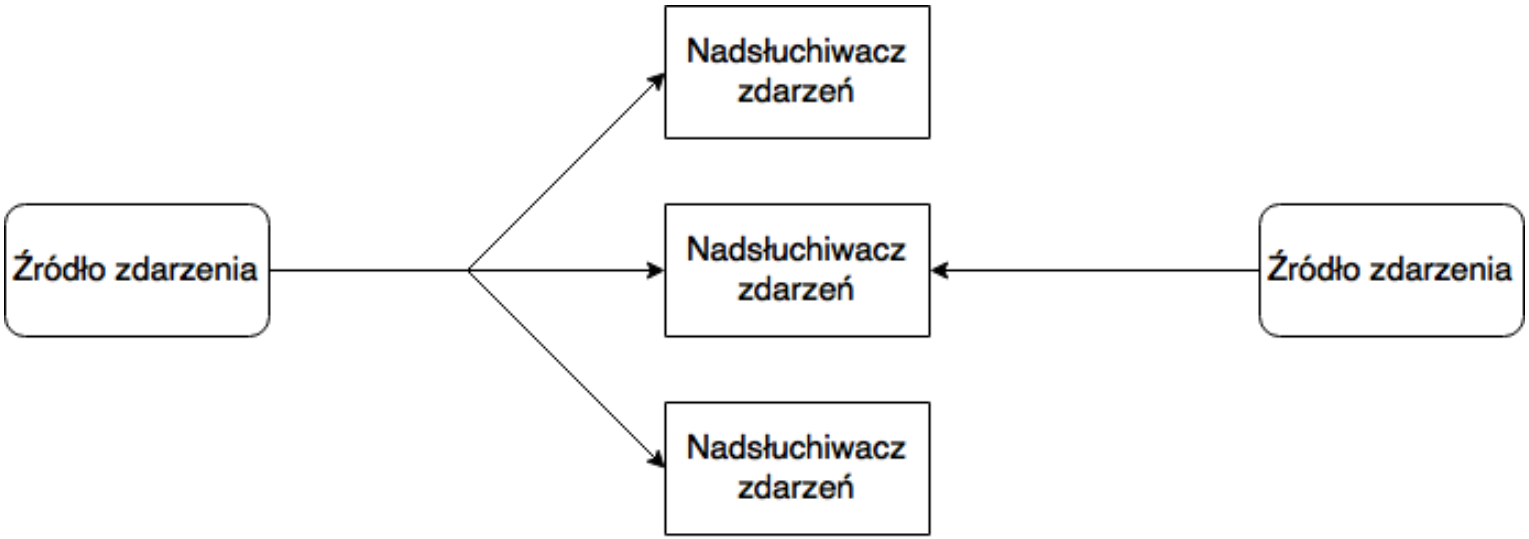
Projekt „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca” współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Zadanie 10 pn. „Modyfikacja programów studiów na kierunkach prowadzonych przez Wydział Matematyki i Nauk Informacyjnych”, realizowane w ramach projektu „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca”, współfinansowanego jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Komunikacja z użytkownikiem

- Tworząc interfejs graficzny dopuszczamy, aby w działanie naszych algorytmów ingerował nieprzewidywalny czynnik - użytkownik.
- Użytkownik, za pomocą dostępnych narzędzi - klawiatury, myszki, sensorów dotykowych - może w nieokreślonym momencie wpływać na działanie programu.
- Java udostępnia nam narzędzia do radzenia sobie z użytkownikami. Są to mechanizmy obsługi zdarzeń.

Mechanizm obsługi zdarzeń



Rysunek 1: Elastyczny mechanizm obsługi zdarzeń w Javie

- Istnieją klasy, służące głównie do obsługi urządzeń peryferyjnych, które mogą generować zdarzenia i wysyłać informacje o ich wystąpieniu do wielu odbiorców.
- Dowolna klasa może nasłuchiwać wystąpienia zdarzeń, o ile implementuje odpowiedni interfejs i jest zarejestrowana w źródle zdarzeń.

Przykład obsługi zdarzeń

- Najczęstszym przypadkiem obsługi zdarzeń jest reakcja na naciśnięcie przycisku.
- Generuje ono zdarzenie `ActionEvent`, które jest przekazywane do zarejestrowanych obiektów.
- Zarejestrowany obiekt musi implementować interfejs `ActionListener`.

Źródło zdarzeń

```
1  JButton button = new JButton("Selfdestruction");
2  button.addActionListener(new MyListener ());
```

Nasłuchiwanie zdarzeń

```
1
2  class MyListener implements ActionListener {
3    public void actionPerformed(ActionEvent event){
4       myHeadquarter.destroy();
5    }
6  }
```

Klasa `ActionEvent`

- Klasa `ActionEvent` nadpisuje klasy z bibliotek util i AWT `EventObject` i `AWTEvent`.
- Dzięki temu można uzyskać dostęp do kontrolki, która wywołała zdarzenie wywołując metodę `getSource`.
- Można także odczytać jaka komenda została przesłana metodą `getActionCommand`.
  - domyślnie jest to etykieta komponentu

Interfejs `ActionListener`

- Obiekt implementujący`ActionListener` może nasłuchiwać:
  - wciśnięcia kontrolki przycisku,
  - wyboru elementu z listy za pomocą dwukrotnego kliknięcia,
  - wyboru elementów menu,
  - wciśnięcia klawisza Enter w polu tekstowym,
  - upływu określonego okresu czasu dla komponentu `Timer`.
- Istnieją też inne interfejsy pozwalające na nasłuch innych zdarzeń.

Interfejsy nasłuchu

- Można nasłuchiwać:
  - Zdarzeń dotyczących okna
    - Zmniejszanie, powiększanie, zmiana fokusu.
    - `WindowListener`, `WindowFocusListener`, `WindowStateListener`
  - Wybóru elementów z listy
    - `ItemListener`
  - Zdarzeń dotyczących myszy
    - Wciśniecie przycisku, ruch, ruch kółka, ciągnięcie
    - `MouseListener`, `MouseMotionListener`, `MouseWheelListener`
  - Zmiany określonego atrybutu obiektu
    - `PropertyChangeListener`

Nowoczesne tworzenie elementów nasłuchowych

Klasy abstrakcyjne i wyrażenia lambda, pozwalają wyeliminować konieczność definiowania klas nasłuchujących. Definiuje się tylko wykonywaną akcję.

Nadpisywanie nasłuchu dla okna

- Załóżmy, że chcemy nadpisać interfejs `WindowStateListener`, aby zareagować na zamknięcie okna.
- W tym celu powinniśmy nadpisać metodę `windowClosing`, jednakże interfejs zawiera szereg metod, które także musimy zaimplementować

Interfejs `WindowStateListener`

```
1  public interface WindowListener{
2   void windowOpened(WindowEvent e);
3   void windowClosing(WindowEvent e);
4   void windowClosed(WindowEvent e);
5   void windowIconified(WindowEvent e);
6   void windowDeiconified(WindowEvent e);
7   void windowActivated(WindowEvent e);
8   void windowDeactivated(WindowEvent e);
9  }
```

- W celu uniknięcia implementacji pozostałych metod można wykorzystać *klasy adaptacyjne*.

Klasa adaptacyjna

- Klasa adaptacyjna zawiera puste implementacje metod interfejsu.
- Wystarczy, że nadpiszemy metodę, której potrzebujemy.
- W przypadku interfejsu `WindowStateListener` jest to klasa `WindowAdapter`

Nadpisanie klasy `WindowAdapter`

```
1  class Terminator extends WindowAdapter{
2    public void windowClosing(WindowEvent e){
3      System.exit(0);
4    }
5  }
```

- Zauważmy, że można by osiągnąć to samo, gdyby interfejs `WindowStateListener` miał metody domyślne.

Akcje

- W przyjaznym interfejsie użytkownika tę samą akcję możemy wywołać na kilka sposobów.
  - Wciśnięcie przycisku.
  - Wybranie pozycji z menu.
  - Wciśnięcie skrótu klawiszowego.
- Oczekiwalibyśmy, że każdy z tych sposobów wywołania spowoduje takie same konsekwencje.
- Java oferuje mechanizm Akcji pozwalających na zdefiniowanie tego samego zachowania dla różnych wywołań Akcji.
- Stosuje się go poprzez nadpisanie interfejsu `Action`, który rozszerza `ActionListener`.

Interfejs Action

Interfejs Action

```
1  Interface Action extends ActionListener{
2    void actionPerformed(ActionEvent event)
3    void setEnabled(boolean b)
4    boolean isEnabled()
5    void addPropertyChangeListener(PropertyChangeListener listener)
6    void removePropertyChangeListener(PropertyChangeListener listener)
7    void putValue(String key, Object value)
8    Object getValue(String key)
9  }
```

- Interfejs zawiera metodę `actionPerformed`. która działa tak samo jak w przypadku `ActionListener`.
- Dodatkowo zawiera metody ustalające czy akcja może zostać przeprowadzona `setEnabled` i `isEnabled`.
- Metody obsługujące `PropertyChangeListener` służą do propagacji zmian wartości.
- Metody `putValue` i `getValue` służą do określania parametrów akcji.

Parametry akcji

NAME Nazwa akcji wyświetlana na przyciskach i elementach menu

SMALL_ICON Ikona wyświetlana na przyciskach, pasku narzędzi i elementach menu

SHORT_DESCRIPTION Krótki opis wyświetlany w etykiecie narzędzia

MNEMONIC_KEY Skrót akcji (uruchamiany ALT+skrót)

Dodawanie akcji

- Akcje można definiować nadpisując adapter `AbstractAction`
- Definiujemy akcję określając jej ikonę i etykietę.

```
1   AbstractAction miniAction = MiniAction("Faculty of Mathematics
        and Information Science",new
        ImageIcon("LogoMiNIsmall.png"));
```

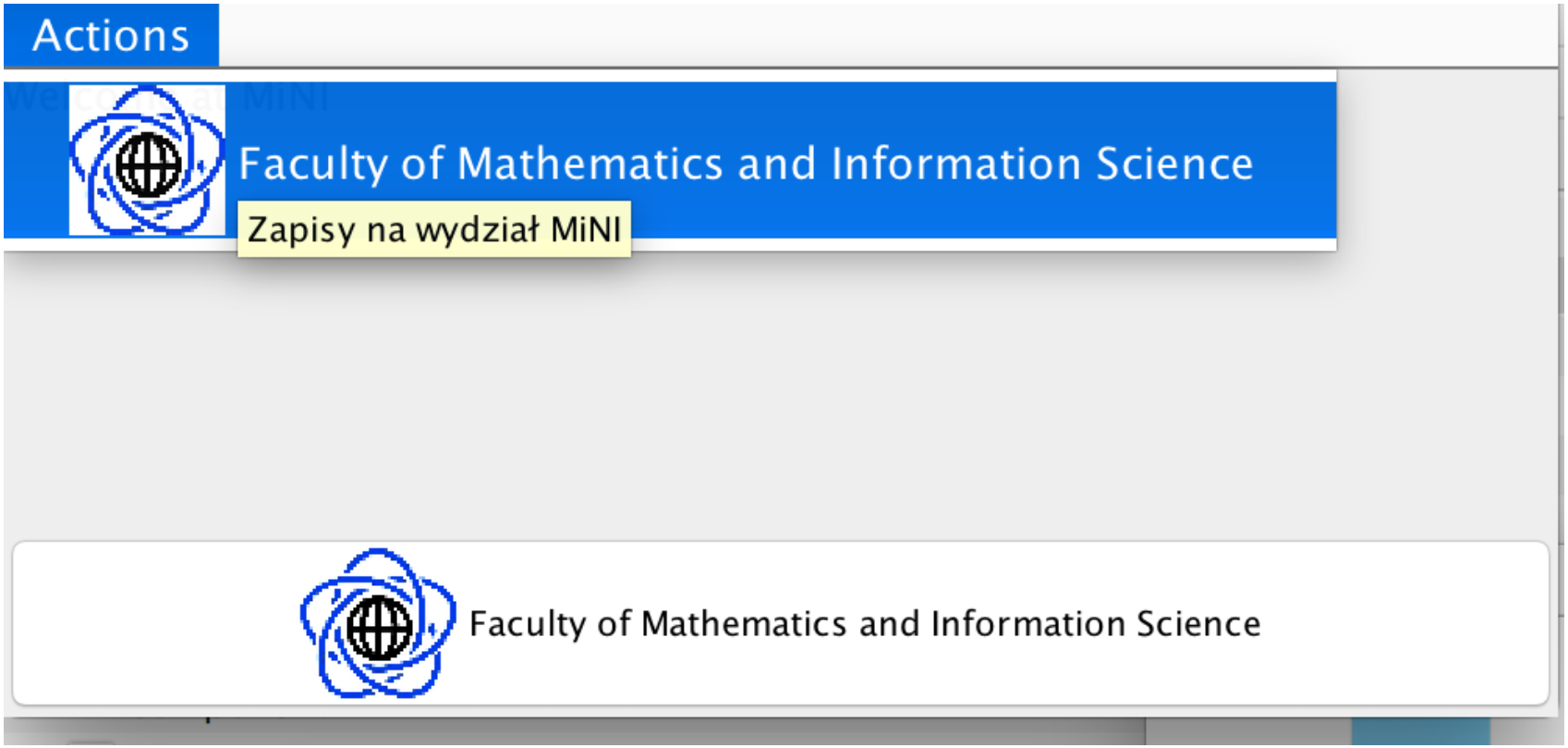
- Zdefiniowaną akcję możemy wykorzystać do tworzenia przycisków

```
1   JButton miniButton = new JButton(miniAction);
```

- Możemy także tworzyć elementy menu

```
1   MenuItem miniMenuItem= new MenuItem(miniAction);
```

Akcje przypisane do komponentów



Rysunek 2: Program wykorzystujący akcję `MiniAction`

Bibliografia