# Obsługa zdarzeń

## 1. Definicja i idea

Każda z aplikacji zmienia swój stan (reaguje) pod wpływem zdarzeń. Mogą to być zdarzenia generowane przez urządzenia wejściowe (mysz, klawiatura), jak również poszczególne składniki programu (okna, timer'y itp.), czy wreszcie elementy systemu operacyjnego, pod kontrolną którego pracuje. Od programisty zależy czy i w jaki sposób, poszczególne zdarzenia zostaną obsłużone.

Aby korzystać z obsługi zdarzeń, należy zaimportować pakiet awt.:

import.java.awt.\*;

import.java.awt.event.\*;

Istnieją dwa modele obsługi zdarzeń. Starszy (tzw. tradycyjny), który obecnie wychodzi już z użycia polega na dziedziczeniu. Oznacza to, że zdarzenie jest obsługiwane przez metodę action (zdarzenia generowane przez użytkownika) lub handleEvent (zdarzenia generowane przez komponenty graficzne). Może także zostać przekazane do klasy nadrzędnej (nadklasy). To już w praktyce nie jest używane.

Obecnie w użyciu jest nowy model obsługi zdarzeń, oparty na tzw. delegacji (model delegacyjny). Polega on na tym, że do obsługi zdarzenia mogą być delegowane dowolne obiekty, które implementują odpowiedni interfejs nasłuchujący (tzw. zarządca zdarzeń).

W tabeli poniżej znajduję się zestawienie poszczególnych interfejsów zarządzających zdarzeniami.

Nazwa interfejsu	Opis
ActionListener	obsługuje zdarzenia generowane przez użytkownika na rzecz
	danego składnika interfejsu (np. kliknięcie przycisku)
AdjustmentListener	obsługuje zdarzenie jako zmianę stanu składnika (np. przesuwanie suwaka w polu tekstowym)
FocusListener	obsługuje zdarzenie od przejścia składnika w stan nieaktywny
ItemListener	obsługuje zdarzenie od np. zaznaczenia pola wyboru
KeyListener	obsługuje zdarzenie np. od wpisywania tekstu z klawiatury
MouseListener	obsługuje zdarzenie od naciśnięcia klawiszy myszy
MouseMotionListener	obsługuje zdarzenie od przesuwania wskaźnika myszy nad
	danym składnikiem
WindowListener	obsługuje zdarzenie od okna np. minimalizacja,
	maksymalizacja, przesunięcie, zamknięcie

## 2. Przykład obsługi zdarzenia - obsługa myszy

Jeżeli chcemy implementować delegacyjną metodę obsługi zdarzeń, musimy na początek zadeklarować klasę. Będzie ona przeznaczona do obsługi określonego zdarzenia:

```
Class ObslugaMyszy implements MouseListener,
MouseMotionListener {

public void mouseClicked (MouseEvent evt) {}

public void mouseDragged (MouseEvent evt) {}

public void mouseEntered (MouseEvent evt) {}

public void mouseExited (MouseEvent evt) {}

public void mouseMoved (MouseEvent evt) {}

public void mousePressed (MouseEvent evt) {}

public void mouseReleased (MouseEvent evt) {}

}
```

Jeśli zamierzamy użyć funkcji obsługi zdarzeń, więcej niż jednej kategorii (np. tylko funkcji mousePressed (z klasy MouseListener) oraz mouseDragged ( z klasy MouseMotionListener)) wymagane jest, aby deklarować pozostałe (nawet te, których nie zamierzamy używać - mouseMoved) funkcje w wykorzystywanej klasie nasłuchującej (w tym przypadku jest to klasa MouseMotionListener)

Kolejną czynnością jaką wykonamy, będzie zarejestrowanie (przypisanie) klasy nasłuchującej, dla danego obiektu:

```
//tworzymy nową instancję klasy ObslugaMyszy
ObslugaMyszy om = new ObslugaMyszy ();
//przypisanie dla interfejsów MouseListener
addMouseListener (ObslugaMyszy);
// przypisanie dla interfejsów MouseMotionListener
addMouseMotionListener (ObslugaMyszy);
```

Ostatnim krokiem, jest zaimplementowanie metody obsługi zdarzenia.

```
public void mouseDragged (MouseEvent evt) {
  int x = evt.getX();
  int y = evt.getY();
  g.drawLine(xOld, yOld, x, y);
  xOld = x;
  yOld = y;
  }
```

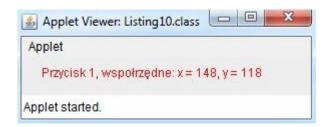
W powyższym przykładzie, zdarzeniu mouseDragged (przeciągnięcie wskaźnika myszy) przypisaliśmy funkcję rysowania linii.

#### Listing 10) Prezentacja obsługi myszy:

```
public class ObslugaMyszy implements MouseListener, MouseMotionListener {
private MojPanel mp;
public ObslugaMyszy(MojPanel mp) {
       this.mp = mp;
@Override
public void mouseClicked(MouseEvent evt) {
       int button = evt.getButton();
       String tekst;
       switch(button){
              case MouseEvent.BUTTON1 : tekst = "Przycisk 1, "; break;
              case MouseEvent.BUTTON2 : tekst = "Przycisk 2, "; break;
case MouseEvent.BUTTON3 : tekst = "Przycisk 3, "; break;
              default: tekst = "Przycisk nieznany, ";
       }
       tekst += "wspolrzędne: x = " + evt.getX() + ", y = " +
          evt.getY();
       mp.setTekst(tekst);
       mp.repaint();
}
@Override
public void mouseDragged(MouseEvent evt) {}
@Override
public void mouseEntered(MouseEvent evt) {}
@Override
public void mouseExited(MouseEvent evt) {}
@Override
public void mouseMoved(MouseEvent evt) {}
@Override
public void mousePressed(MouseEvent evt) {}
@Override
public void mouseReleased(MouseEvent evt) {}
public class MojPanel extends JComponent {
private String tekst;
private ObslugaMyszy om;
public MojPanel() {
       setPreferredSize(new Dimension(300,300));
       tekst = "";
       om = new ObslugaMyszy(this);
       addMouseListener(om);
       addMouseMotionListener(om);
       repaint();
```

```
public void setTekst(String tekst) {
      this.tekst = tekst;
@Override
public void paintComponent(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      g2.setColor(Color.RED);
      g2.drawString(tekst, 20, 20);
public class Listing10 extends JApplet {
private MojPanel mp;
public Listing10(){
       mp = new MojPanel();
        setLayout(new BorderLayout());
        add(mp, BorderLayout.CENTER);
public static void main(String[] args) {
      JFrame window = new JFrame();
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setTitle("Listing10");
        window.setContentPane(new Listing10());
        window.pack();
        window.setLocationRelativeTo(null);
        window.setVisible(true);
```

Wynik działania apletu:



## 3. Przykład obsługi zdarzenia - obsługa klawiatury

Jeżeli chcemy implementować delegacyjną metodę obsługi zdarzeń, musimy na początek zadeklarować klasę. Będzie ona przeznaczona do obsługi określonego zdarzenia:

```
Class ObslugaKlawiatury implements KeyListener {
public void keyPressed (KeyEvent evt) {}
public void keyReleased (KeyEvent evt) {}
    public void keyTyped (KeyEvent evt) {}
}
```

#### Listing 11) Prezentacja obsługi klawiatury:

```
public class ObslugaKlawiatury implements KeyListener {
private MojPanel mp;
public ObslugaKlawiatury(MojPanel mp){
      this.mp = mp;
}
@Override
public void keyPressed(KeyEvent evt) {
      String tekst = mp.getTekst();
      tekst += evt.getKeyChar();
      mp.setTekst(tekst);
      mp.repaint();
@Override
public void keyReleased(KeyEvent evt) {}
@Override
public void keyTyped(KeyEvent evt) {}
public class MojPanel extends JComponent {
private JTextField typingArea;
private String tekst;
private ObslugaKlawiatury ok;
public MojPanel() {
      setPreferredSize(new Dimension(300,300));
      tekst = "";
      ok = new ObslugaKlawiatury(this);
      typingArea = new JTextField(20);
             typingArea.addKeyListener(ok);
             add(typingArea);
      repaint();
public String getTekst(){
      return tekst;
public void setTekst(String tekst) {
      this.tekst = tekst;
@Override
public void paintComponent(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      g2.setColor(Color.RED);
      g2.drawString(tekst, 20, 20);
```

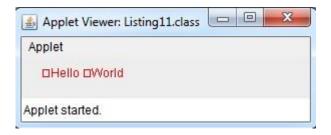
```
public class Listing11 extends JApplet {
private MojPanel mp;

public Listing11(){
    mp = new MojPanel();

    setLayout(new BorderLayout());
    add(mp, BorderLayout.CENTER);
}

public static void main(String[] args) {
    JFrame window = new JFrame();
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setTitle("Listing11");
        window.setContentPane(new Listing11());
        window.pack();
        window.setLocationRelativeTo(null);
        window.setVisible(true);
}
```

Wynik działania apletu:



### 4. Inne rodzaje zdarzeń

Oprócz klawiatury i myszy, zdarzenia mogą generować także inne źródła:

- ActionListener obsługuje zdarzenia generowane przez użytkownika na rzecz danego składnika,
- AdjustmentListener obsługuje zdarzenia generowane w momencie zmiany stanu składnika (np. uzytkownik przesunie suwak w polu teksotwym),
- Focuslistener obsługuje zdarzenia generowane w momencie, kiedy takie składniki, jak np. pole tekstowe stają się lub przestają być aktywne,
- Itemlistener obsługuje zdarzenia generowane wówczas, gdy np. zaznaczane jest pole wyboru,
- WindowListener obsługuje zdarzenia generowane przez okno (Jwindow, Jframe), takie jak maksymalizacja, minimalizacja, przesunięcie czy zamknięcie okna.