POLIMORFIZM, INTERFEJSY, PROGRAMOWANIE ZDARZENIOWE
Interfejsy

- Są definiowane za pomocą słowa **interface**
- Nie zawierają definicji metod, lecz tylko ich deklaracje publiczne (**public**)
- Nie zawierają składowych typu dane
- Zawierają definicje stałych finalnych (**public static final**)
- Zastępują wielokrotne dziedziczenie w C++
- Klasa implementuje interfejsy (**implements**) tzn. musi zawierać definicje metod implementowanego interfejsu

Przykład

```
package Punkt2;                    //plik Figura.java

public interface Figura
{
    String figura ="figura:  ";    //public static final
     void rysuj();                 //public
}
```

```
package Punkt2;                    //plik punkt2.java

import java.lang.*;
class Punkt implements Figura
{
  int x, y;

   Punkt(int wspX, int wspY)
        {    x = wspX;  y = wspY;}
 void zmien(int wspX, int wspY)
        {    x = wspX;  y = wspY;}
```

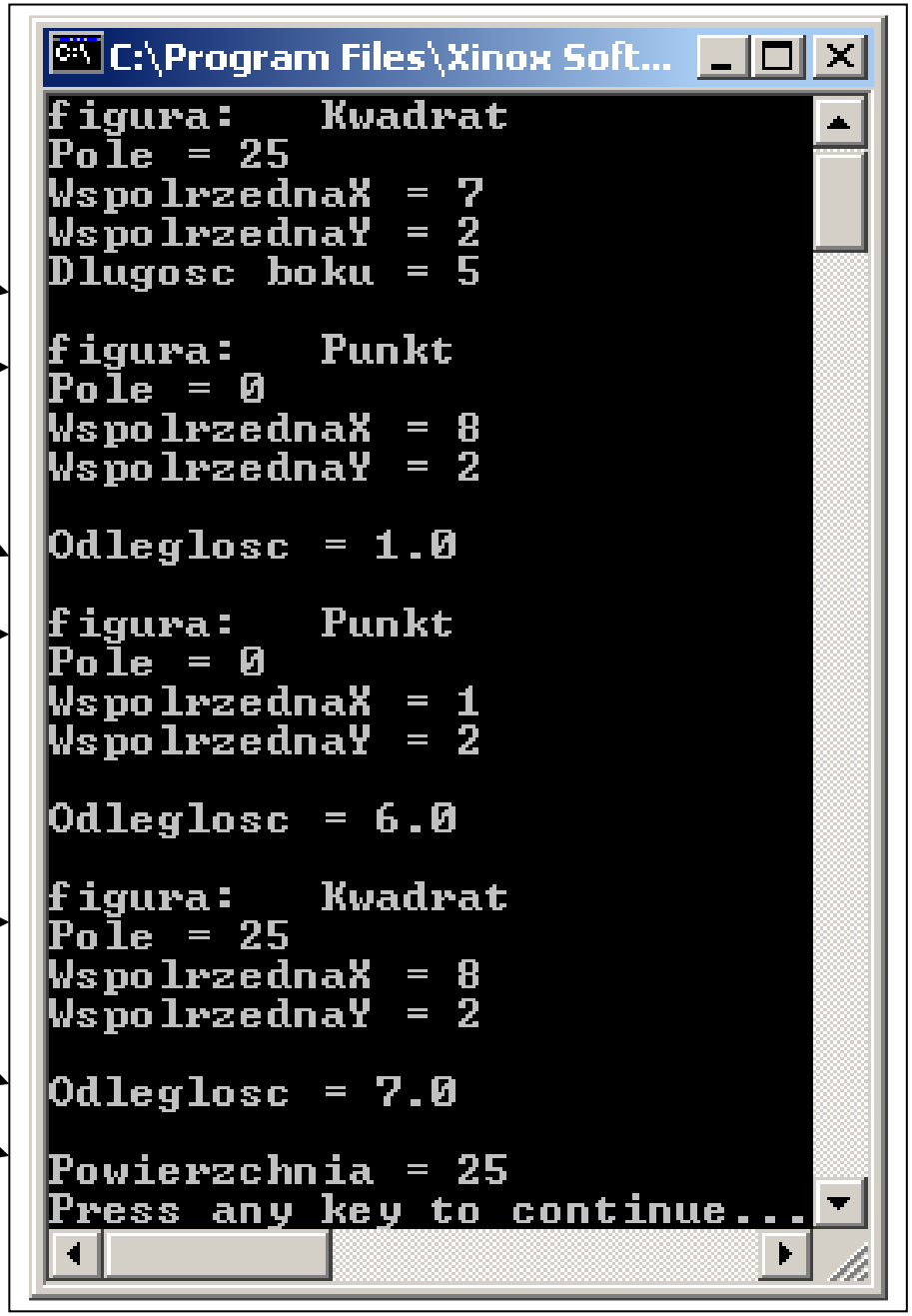
```
 int podajX()
        {    return x;  }
 int podajY()
        {    return y;  }
 void przesun(int dx, int dy)
        {    x+=dx;  y+=dy;  }
 double odleglosc(Punkt p)
       {      return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));  }
 int pole ()
       {      return 0;}
 public void rysuj()                    //implementacja metody rysuj
       {
              System.out.println('\n'+figura            +" Punkt");
       }
}
```

```
class Kwadrat extends Punkt
{  int dlugosc;
   Kwadrat(int wspX, int wspY, int dlugosc_)
       {      super(wspX,wspY);
              dlugosc= dlugosc_;
       }
 int podajDl()
       {      return dlugosc;  }

 int pole()
       {      return dlugosc*dlugosc;  }
```

```
public void rysuj()                    //implementacja metody rysuj
                                  //figura – publiczna stała finalna z interfejsu Figura
      {     System.out.println('\n'+figura+ " Kwadrat");  }
}
```

```
public class punkt2                    //klasa publiczna, nieabstrakcyjna, niefinalna
{
     //metoda typu static może być wywołana w metodzie main bez tworzenia
     //obiektu typu punkt2

static void wyswietl(Punkt p)
      {
       //dzięki polimorfizmowi będzie wywołana metoda  pole  oraz
      // dzięki implementacji będzie wywołana metoda  rysuj
      //albo klasy Punkt albo Kwadrat
      // w zależności od typu aktualnego obiektu podstawionego za p

            p.rysuj();
            System.out.println("Pole = "+ p.pole());
            System.out.println("WspolrzednaX = "+ p.podajX());
            System.out.println("WspolrzednaY = "+ p.podajY());
      }
```

```
public static void main (String[] args)
 {
    Kwadrat k1 = new Kwadrat(7,2,5);
    wyswietl(k1);
    System.out.println("Dlugosc boku = "     + k1.podajDl());


    Punkt p2 = new Punkt(8,2);
    wyswietl(p2);
    System.out.println("\nOdleglosc = " + p2.odleglosc(k1));


   p2.zmien(1,2);
    wyswietl(p2);
    System.out.println("\nOdleglosc = " + p2.odleglosc(k1));


    k1.przesun(1,0);
    wyswietl(k1);
    System.out.println("\nOdleglosc = "+ k1.odleglosc(p2));

   System.out.println("\nPowierzchnia = "+ k1.pole());
 }
}
```



Programowanie wizualne- pakiet Swing

1. Główny obiekt interfejsu użytkownika - obiekt klasy JFrame

1.1. Przykład prostej aplikacji

- Zdefiniowanie klasy dziedziczącej po klasie **JFrame** z pakietu **Swing** (lub **JWindow**)

```
public class Prosta_Aplikacja2 extends JFrame
        {  //.....
        }
```

- Wywołania konstruktora klasy bazowej dla zapewnienia jej poprawnej inicjalizacji: **super()** (dwa konstruktory: **JFrame()** - okno nie posiada wtedy tytułu oraz **JFrame(String)** - okno posiada wtedy tytuł podany w parametrze konstruktora)
- Ustawienia rozmiaru okna ramki w pikselach za pomocą metody **setSize()**
- Obsługa zamykania okna
- Wyświetlenie ramki na ekranie metodą show() lub **setVisible(true)**

Ramka posiada przyciski Minimalizuj, Maksymalizuj oraz Zamknij zlokalizowane na pasku tytułu. W przypadku Javy domyślne zamknięcie ramki nie powoduje zamknięcia aplikacji. Aby to zmienić, należy wywołać metodę **setDefaultCloseOperation()** zawierającą jeden z czterech parametrów wywołania:

- ✓ **EXIT_ON_CLOSE** – zamyka aplikacje po zamknięciu ramki
- ✓ **DISPOSE_ON_CLOSE** – zamyka ramkę, usuwa obiekt ramki, ale pozostawia pracującą aplikację,
- ✓ **DO_NOTHING_ON_CLOSE** – pozostawia ramkę otwartą i kontynuuje pracę aplikacji
- ✓ **HIDE_ON_CLOSE** – zamyka ramkę pozostawiając pracującą aplikację.


```
import javax.swing.JFrame;
public class Prosta_Aplikacja2 extends JFrame
{
     public Prosta_Aplikacja2()
         {
           super("Ramka 1");
           setSize(250,50);
           setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
           setVisible(true);
          }

   public static void main(String[] arg)
    {
       Prosta_Aplikacja2 pr= new Prosta_Aplikacja2();
    }
}
```

1.2. Zamykanie okna aplikacji za pomocą bezpośredniej obsługi zdarzenia zamykania okna (zamiast wykorzystania metody setDefaultCloseOperation())

Czynności:

- Utworzyć obiekt, który będzie monitorował stan okna

- Implementować za pomocą klasy pochodnej po JFrame interfejs WindowsListener, który będzie obsługiwał wszystkie możliwe zmiany stanu okna za pomocą implementowanych metod
  - ✓ windowClosed() – okno jest zamknięte
  - ✓ windowClosing() – okno jest zamykane
  - ✓ windowOpen() – okno jest widoczne
  - ✓ windowIconified() – okno zminimalizowane
  - ✓ windowDeiconified() – okno zmaksymalizowane
  - ✓ windowActivated() – okno powiązane z tym obiektem staje się aktywne i może przyjmować dane z klawiatury
  - ✓ windowDeactivated() - okno powiązane z tym obiektem staje się nieaktywne i nie może przyjmować dane z klawiatury

- lub wykorzystać klasę WindowAdapter z pakietu awt, która ma zaimplementowane puste metody jak dla interfejsu WindowListener. Należy wykonać klasę dziedziczącą po WindowAdapter przedefiniowując wybraną metodę wg następującego przykładu

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Zamykanie_Aplikacji extends WindowAdapter
 {
      public void windowClosing (WindowEvent e)
          { System.exit(0);}
 }

public class Prosta_Aplikacja3 extends JFrame
 {
     public Prosta_Aplikacja3()
      { super ("Ramka 2");
        setSize(250,50);
        Zamykanie_Aplikacji wyjscie = new Zamykanie_Aplikacji();
        addWindowListener(wyjscie);
        //wstawienie obiektu monitorującego zamykanie okna wyjscie do okna aplikacji
    }


  public static void main (String[] arg)
    {
      Prosta_Aplikacja3 p2= new Prosta_Aplikacja3();
      p2.setVsible(true);
    }
```

2. Obsługa zdarzeń

Jest to reakcja na działania użytkownika programu. Obsługa zdarzeń realizowana jest za pomocą implementowania interfejsów, zwanych zarządcami zdarzeń.

Interfejsy- zarządcy obsługi zdarzeń

- **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

Wiązanie składnika z obsługą zdarzeń

Aby powiązać składnik interfesju z odpowiednim zarządcą zdarzeń, należy wykorzystać następujące metody:

- addActionListener() dla JButton, JCheckBox, JComboBox, JTextField, JRadioButton
- addAdjustmentListene() dla JScroolBar
- addFocusListener() dla wszystkich składników pakietu Swing
- addItemListener() dla JButton, JCheckBox, JComboBox, JTextField, JRadioButton
- addKeyListener() dla wszystkich składników pakietu Swing
- addMouseListener() dla wszystkich składników pakietu Swing
- addMouseMotionListener() wszystkich składników pakietu Swing
- addWindowListener() wszystkich obiektów typu JFrame oraz JWindow

Metody te muszą być zastosowane przez wstawieniem składnika do obiektu typu JPanel.

Metody obsługujące zdarzenia

Kiedy dana klasa implementuje interfejs, musi ona obsługiwać zdarzenia za pomocą metody, która jest wtedy wywoływana automatycznie, natomiast w programie trzeba ją odpowiednio przedefiniować, realizując sposób obsługi zdarzeń.

- Interfes ActionListener

  public void actionPerformed (ActionEvent evt)

  { //kod obsługi zdarzeń }

- Interfes AjustmentListener

  public void adjustmentValueChange (AdjustmentEvent evt)

  {//kod obsługi zdarzeń }

- Interfes FocusListener

  public void focusGained (FocusEvent evt) { //kod obsługi zdarzeń }

  public void focusLost (FocusEvent evt) { //kod obsługi zdarzeń }

- Interfes ItemListener

  public void itemStateChange (ItemEvent evt) { //kod obsługi zdarzeń }

- Interfes KeyListener

  public void keyPressed (KeyEvent evt) { //kod obsługi zdarzeń }

  public void keyReleased (KeyEvent evt) { //kod obsługi zdarzeń }

  public void keyTyped (KeyEvent evt) { //kod obsługi zdarzeń }

- Interfes MouseListener
  - **public void** mouseClicked (MouseEvent evt) { //kod obsługi zdarzeń }
  - **public void** mouseEntered (MouseEvent evt) { //kod obsługi zdarzeń }
  - **public void** mouseExited (MouseEvent evt) { //kod obsługi zdarzeń }
  - **public void** mousePressed (MouseEvent evt) { //kod obsługi zdarzeń }
  - **public void** mouseReleased (MouseEvent evt) { //kod obsługi zdarzeń }
- Interfes MouseMotionListener
  - **public void** mouseDragged (MouseEvent evt) { //kod obsługi zdarzeń }
  - **public void** mouseMoved (MouseEvent evt) { //kod obsługi zdarzeń }
- Interfes WindowListener
  - **public void** windowActivated (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowClosed (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowClosing (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowDeactivated (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowDeiconfied (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowIconfied (WindowEvent evt) { //kod obsługi zdarzeń }
  - **public void** windowOpened (WindowEvent evt) { //kod obsługi zdarzeń }

Przykład 2 – Obsługa zdarzeń – naciskanie klawiszy KeyListener

```
class Punkt
{ protected int x, y;

  public Punkt(int wspX, int wspY)
     {x = wspX; y = wspY;}

  public int podajX()
   { return x;}

  public int podajY()
   { return y;}

  public double odleglosc(Punkt p)
   { return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y)); }

  public void rysuj(Graphics g)
    {  Graphics2D g2D= (Graphics2D)g;
       Color pedzel =new Color(255,0,0);
       g2D.setColor(pedzel);
       g2D.fillOval(x,y,5,5);
     }

  public void przesun(int dx, int dy, int a, int b)
   {   x+=dx; y+=dy;
      if (x>a || x<1) x=5;
       if (y>b || y<1) y=2;
   }
  }
```

```
class Kwadrat extends Punkt
{
   protected int dlugosc;

  public Kwadrat(int wspX, int wspY,int dl)
   {
      super(wspX,wspY);
      dlugosc=dl;
   }

  public void przesun(int dx, int dy, int a, int b)
    {
      super.przesun(dx,dy, a, b);      //przesuwanie
      dlugosc+=dx+dy;                  //i skalowanie
      if (dlugosc > 30 || dlugosc <10)
          dlugosc=20;
     }

  public void rysuj(Graphics g)
    {
     Graphics2D g2D=(Graphics2D)g;
     Color pedzel=new Color(0,255,0);
     g2D.setColor(pedzel);
     g2D.fillRect(x,y,dlugosc,dlugosc);
    }
 }
```

```
class Figury
{  protected int N=6;
   protected int biezacy=0;   //wybrana figura do animacji
   protected Punkt figury[];

public Figury()
    {  wypelnij();
       polozenie();
     }

public void wypelnij()
  {  figury =new Punkt[N];
     for (int i=0; i<figury.length; i++)
       { figury[i]=new Punkt(i*22,i*22);
           if (i<figury.length-1)
              figury[++i]= new Kwadrat (i*44,i*44,20); }
   }

public void polozenie() {
      for (int i=0; i<figury.length-1;i++)
       {  boolean p=figury[i] instanceof Kwadrat;
          System.out.println("\n"+
                 p+", ze jestem kwadratem, bo jestem "+figury[i].toString()+
                 " ,X="+figury[i].podajX()+
                 " , Y="+figury[i].podajY()+
                 " , odleglosc="+figury[i].odleglosc(figury[i+1]));
       }
   }
```

```
  public void wybierz_biezaca_figure(int pom)
     { if (!(pom>=0&& pom<N))
          pom=0;
       biezacy=pom;
       System.out.println(pom); }

  public void przesun(int dx, int dy, int maxx, int maxy)
     {      figury[biezacy].przesun(dx, dy, maxx, maxy);    }

  public void rysuj(Graphics g)
     {    for (int i=0; i<figury.length; i++)
            figury[i].rysuj(g);  }          //automatyczne rysowanie albo punktu albo kwadratu
}
```

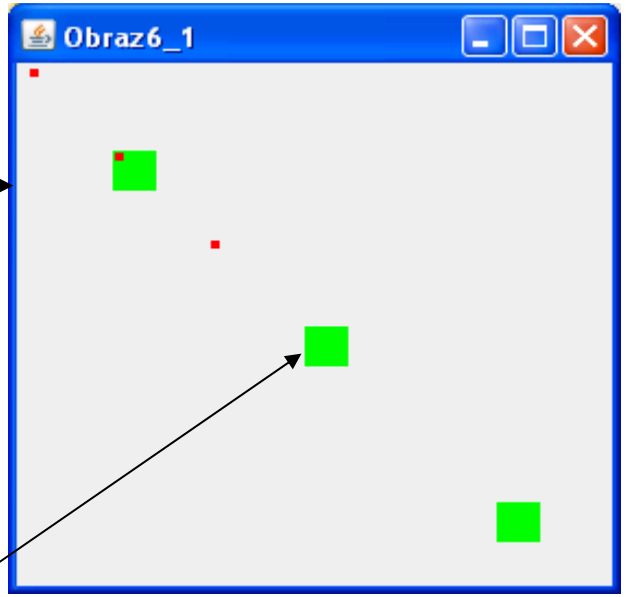
```
class Panel extends JPanel implements KeyListener // klasa dostarczająca zawartość graficzną dla ramki (JFrame)
  { public Figury rys = new Figury();

    public Panel()
      { super();
        addKeyListener(this);
        setFocusable(true); }

   public void paintComponent(Graphics g)  //przedefiniowanie metody  klasy JPanel
    {
      super.paintComponent(g); //dodanie tej linii do programu wywołuje standardowe czyszczenie okna
      rys.rysuj(g);    }              //automatyczne rysowanie albo punktu albo kwadratu
```

```
public void keyPressed (KeyEvent evt)   //reakcja na klawisze strzałek
 //Virtual keys (arrow keys, function keys, etc) - handled with keyPressed() listener.
 { int dx=0, dy=0, zmiana;
   Random r=new Random();
   if (evt.isShiftDown())
           zmiana = r.nextInt(10);
  else  zmiana =r.nextInt(150);
   switch (evt.getKeyCode())
     {  case KeyEvent.VK_LEFT :      dx -=  zmiana;      break;
        case KeyEvent.VK_RIGHT:      dx +=  zmiana;      break;
        case KeyEvent.VK_UP   :      dy -=  zmiana;      break;
        case KeyEvent.VK_DOWN :      dy +=  zmiana;      break;
     }
      rys.przesun(dx,dy, getWidth(),getHeight());
     repaint();   } //odwieżenie widoku okna

public void keyReleased (KeyEvent evt) //metoda pusta, niewykorzystana, dlatego
    { }                                //implementowana jako pusta

public void keyTyped (KeyEvent evt)    //obsługa naciskanych klawiszy alfanumerycznych
  //Characters (a, A, #, ...) - handled in the keyTyped() listener.
 { char znak = evt.getKeyChar();      //reakcja na przycisku na naciśnięcie klawisza
    int pom=(int)znak-48;             //obliczenie indeksu figury w tablicy figury z kody ASCCI klawisza
    rys.wybierz_biezaca_figure(pom);  //zapamiętanie numeru wybranej figury
    System.out.println(biezacy);
    repaint();  }                     //odwieżenie widoku okna
```

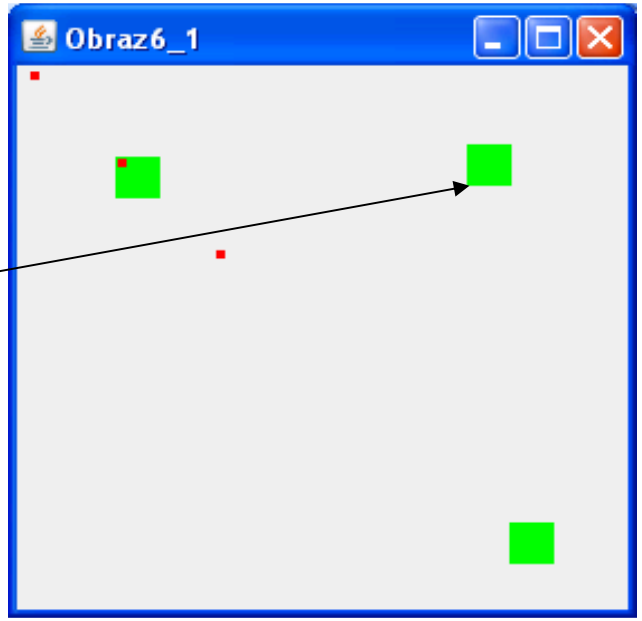
implementacja metod Interfejsu **KeyListener**

```
public class Obraz6_1  extends JFrame
{
  protected Panel obraz;

  public Obraz6_1()
  {
    super ("Obraz6_1");
    setSize(400,400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    obraz=new Panel();
    setContentPane(rysunek);
    setVisible(true);
   }

public static void main(String args[])
  {
     Obraz6_1 obraz=new Obraz6_1();
   }
}
```

Stan początkowy ekranu po narysowaniu figur



Stan ekranu po narysowaniu figur po naciśnięciu klawisza 4 i przesunięciu wybranej figury klawiszami strzałek w nowe położenie – reakcja figur na zdarzenie od klawiszy funkcyjnych i zwykłych.



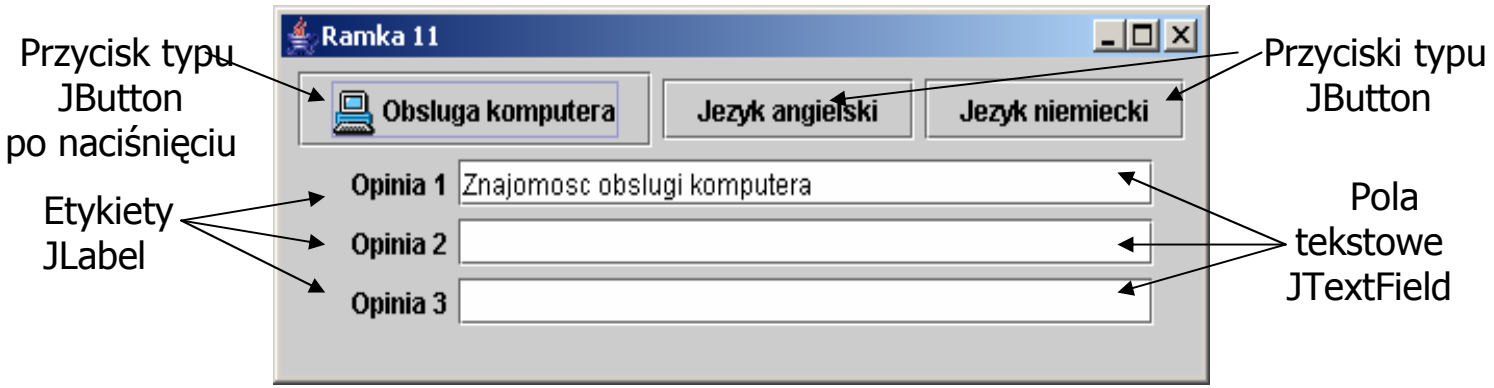
3. Tworzenie składników interfejsu użytkownika

- Dowolny składnik musi być dodany do kontenera (panel-obiekt klasy JPanel) za pomocą jego metody panel.add(skladnik), i po wywołaniu metody setContentPane(panel) wyświetlony na ekranie po wyświetleniu kontenera.
- Obsługa zdarzenia typu **Action** (JButton, JCheckBox, JComboBox, JTextField, JRadioButtion): naciśnięcie klawisza i wyświetlenie napisu w polu tekstowym.

Przyciski typu JButton

Konstruktory:

- ✓ JButton(String) - przycisk z etykietą tekstową
- ✓ JButton(Icon) - przycisk w postaci ikony graficznej
- ✓ JButton(String, Icon) – tworzy przycisk z etykietą tekstową i ikoną
- ✓



```
import javax.swing.*;
import java.util.*;
import java.io.*;
import java.lang.*;
import java.awt.event.*;

class Zamykanie_Aplikacji extends WindowAdapter
 {    public void windowClosing (WindowEvent e)
        { System.exit(0); }
 }

class Dane
{ String opinia1, opinia2, opinia3;
   Dane()
   { opinia1=opinia2=opinia3=null;}
}

public class Prosta_Aplikacja12 extends JFrame implements ActionListener
{ ImageIcon o1=new ImageIcon ("comp.blue.gif");
  JButton weopinia1 = new JButton ("Obsluga komputera", o1); //obiekty
  JButton weopinia2 = new JButton("Jezyk angielski");              // do wprowadzania
  JButton weopinia3 = new JButton("Jezyk niemiecki");              // danych
  JTextField wyopinia1=new JTextField(30);       //obiekty do wyświetlania danych
  JTextField wyopinia2=new JTextField(30);       // na ekranie
  JTextField wyopinia3=new JTextField(30);       // związanych z naciśniętymi klawiszami
  Dane dana = new Dane();        //obiekt do zapamiętania danych wprowadzonych w interfejsie gr.
```

```
public Prosta_Aplikacja12()
 {  super("Ramka 11");
    setSize(450,160);
    JPanel panel=new JPanel();
    weopinia1.addActionListener(this); // obiekt monitorujący zdarzenie  typu Action
    weopinia2.addActionListener(this);
    weopinia3.addActionListener(this);

    panel.add(weopinia1);
    panel.add(weopinia2);
    panel.add(weopinia3);
    JLabel eopinia1= new JLabel("    Opinia 1",SwingConstants.RIGHT);
     panel.add(eopinia1);
     panel.add(wyopinia1);
    JLabel eopinia2= new JLabel("    Opinia 2",SwingConstants.RIGHT);
    panel.add(eopinia2);
     panel.add(wyopinia2);
    JLabel eopinia3= new JLabel("    Opinia 3",SwingConstants.RIGHT);
    panel.add(eopinia3);
     panel.add(wyopinia3);

    Zamykanie_Aplikacji wyjscie = new Zamykanie_Aplikacji();
    addWindowListener(wyjscie);

    setContentPane(panel);
 }
```

```
public void actionPerformed (ActionEvent evt)      //metoda obsługująca zdarzenie Action
  {
    Object zrodlo = evt.getSource();
    if (zrodlo==weopinia1)                          //jeśli naciśnięto przycisk ”Obsługa komputera”
      dana.opinia1= new String("Znajomosc obslugi komputera");
   else  if (zrodlo==weopinia2)                     //jeśli naciśnięto przycisk ”Język angielski”
      dana.opinia2= new String("Znajomosc jezyka angielskiego");
    else if (zrodlo==weopinia3)                     //jeśli naciśnięto przycisk ”Język niemiecki”
    dana.opinia3= new String("Znajomosc jezyka niemieckiego");
    wyopinia1.setText(dana.opinia1);                //przekazanie tekstu związanego z naciśniętymi klawiszami
    wyopinia2.setText(dana.opinia2);                // do składników tekstowych JTextField,
    wyopinia3.setText(dana.opinia3);                // które zastosowano do wyświetlenia tekstu
    repaint();
   }


 public static void main(String[] arg)
  {
    Prosta_Aplikacja12 pr= new Prosta_Aplikacja12();
    pr.setVisible(true);
   }
}
```