

DOSTĘP DO METOD I ZMIENNYCH

Dostęp do zmiennych (na podstawie:L.Lemay,R.Cadenhead,Java 2 dla każdego, Helion 2001)

Zakres „widzialności” metody lub zmiennej	public	protected	domyślny	private
Klasa	tak	tak	tak	tak
Pakiet	tak	tak	tak	nie
Inny pakiet	tak	nie	nie	nie
Podklasa, pakiet	tak	tak	tak	nie
Podklasa, inny pakiet	tak	tak	nie	nie

DZIEDZICZENIE

PRZECIĄŻANIE METOD, PRZEDEFINIOWANIE METOD

1) Dziedziczenie

Przykład 1 – dziedziczenie metod w klasie pochodnej

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

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

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

    public int podajX()
        { return x; }

    public int podajY()
        { return y; }

    public void przesun(int dx, int dy)
        { x+=dx;  y+=dy; }

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

```
class Kwadrat extends Punkt //klasa dziedziczy od klasy Punkt
{
    protected int dlugosc;

    Kwadrat(int wspX, int wspY, int dlugosc_)
    {
        super(wspX,wspY); //wywołanie dziedziczonego konstruktora
        dlugosc=dlugosc_;
    } //ponieważ nie ma konstruktora domyślnego (jawnego lub niejawnego konstruktora bez parametrów) w klasie
    // bazowej Punkt-konstruktora domyślnego się nie wywołuje

    int podajDI()
    {
        return dlugosc;
    }

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

```

public class punkt_
{
    public static void main (String[] args)
    {
        Kwadrat k1 = new Kwadrat(7,2,5);
        System.out.println("WspolrzednaX = " + k1.podajX());
        System.out.println("WspolrzednaY = " + k1.podajY());
        System.out.println("Dlugosc boku = " + k1.podajDI());
        Punkt p2 = new Punkt(8,2);
        System.out.println("\nWspolrzednaX = " + p2.podajX());
        System.out.println("WspolrzednaY = " + p2.podajY());
        p2.zmien(1,2);
        System.out.println("\nOdleglosc = " + p2.odleglosc(k1));
        k1.przesun(1,0);
        System.out.println("\nOdleglosc = " + k1.odleglosc(p2));
        System.out.println("\nPowierzchnia = " + k1.pole());
    }
}

```

Te metody są
wywołane dzięki
dziedziczeniu
składowych od klasy
Punkt

```

C:\Program Files\Xinox Soft...
WspolrzednaX = 7
WspolrzednaY = 2
Dlugosc boku = 5

WspolrzednaX = 8
WspolrzednaY = 2

Odleglosc = 6.0
Odleglosc = 7.0

Powierzchnia = 25
Press any key to continue...

```

Obiekt klasy *Kwadrat* posiada metody klasy *Punkt* i swoje własne (*podajDI()*, *pole()* oraz konstruktor *Kwadrat(int, int, int)*). W programie polecenie **super** pozwala wywołać dziedziczony konstruktor klasy *Punkt*. Konstruktory domyślne nie trzeba wywoływać (są wywoływane niejawnie), pozostałe konstruktory dziedziczone trzeba zawsze wywołać za pomocą *super*.

Wywołanie *p2.odleglosc(k1)* zawiera parametr aktualny metody jako referencję do obiektu klasy pochodnej Kwadrat w odniesieniu do typu formalnego tego parametru równego Punkt.

2) Przekładowanie i przekładowanie metod

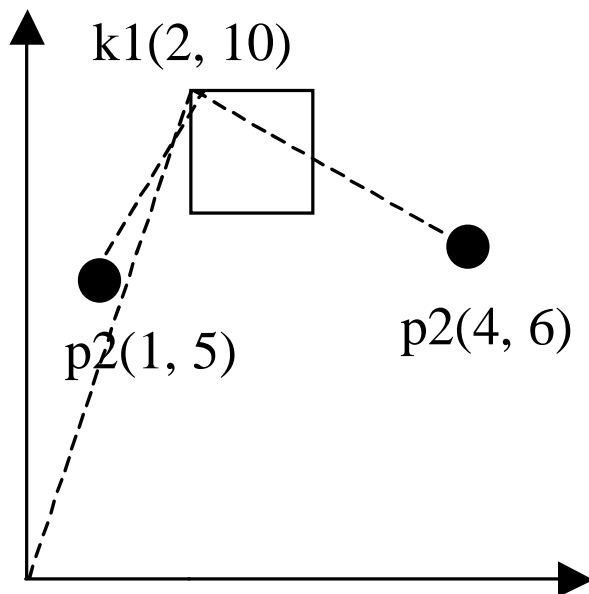
Przekładowanie nazwy metody **met1(...)** dziedziczonej wymaga :

- identycznego nagłowa **met1(...)** w metodzie klasy dziedziczącej
- jest to domyślna metoda klasy dziedziczącej
- metodę bazową w ciele metod klasy dziedziczącej wywołuje się:
super.met1(...)

Przekładowanie nazwy metody **met2(...)** wymaga

- identycznej nazwy **met2**
- różnej liczby parametrów
- różnych typów parametrów, jeśli ich liczba jest identyczna
- typ wyniku zwracanego przez **return** nie przekładowa nazwy metody

Przykład 2 - wykonanie programu prezentującego dziedziczenie, przeddefiniowanie i przeciążenie metod



```
import java.lang.*;
class Punkt
{ protected int x, y;

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

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

    public int podajX()
    { return x; }

    public int podajY()
    { return y; }

    public void przesun(int dx, int dy)
    { x+= dx;
      y+= dy; }
    public int pole ()
    { return 0;}

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

}
```

```
class Kwadrat extends Punkt
```

```
{  
  protected int dlugosc;  
  
  public Kwadrat(int wspX, int wspY, int dlugosc_)  
    { super(wspX, wspY);` //wywołanie dziedziczonego konstruktora od klasy Punkt  
      dlugosc = dlugosc_; }  
  
  public int podajDI()  
    { return dlugosc; }  
  
  public int pole()  
    { return dlugosc*dlugosc; }  
  
  public double odleglosc() //przeciążenie nazwy metody dziedziczonej  
    { return Math.sqrt(x*x+y*y); }  
  
  public double odleglosc(Punkt p) //przedefiniowanie nazwy metody dziedziczonej  
    { return odleglosc() //wywołanie metody przeciążonej  
      + super.odleglosc(p); } //wywołanie metody dziedziczonej  
      // gdy jest przedefiniowana  
  
  public double odleglosc(Punkt p, int i) //przeciążenie nazwy metody dziedziczonej  
    { return super.odleglosc(p); } //wywołanie metody dziedziczonej,  
      // gdy jest przedefiniowana  
}
```

```
public class punkt1
```

```
//klasa publiczna, nieabstrakcyjna, niefinalna
```

```
{  
    public static void main (String[] args)  
    {  
        Kwadrat k1 = new Kwadrat(2,10,1);  
        System.out.println("WspolrzednaX = "+ k1.podajX());  
        System.out.println("WspolrzednaY = "+ k1.podajY());  
        System.out.println("Dlugosc boku = " + k1.podajDl());  
        Punkt p2 = new Punkt(4,6);  
        System.out.println("\nWspolrzednaX = "+ p2.podajX());  
        System.out.println("WspolrzednaY = " + p2.podajY());  
        System.out.println(  
            "\nOdleglosc miedzy punktami\n(4,6),(2,10)=" +  
            p2.odleglosc(k1));  
        p2.zmien(1,5);  
        System.out.println(  
            "\nSuma odleglosci miedzy punktami\n(0,0),(2,10),(1,5)=" +  
            k1.odleglosc(p2));  
        System.out.println(  
            "\nOdleglosc miedzy punktami\n(2,10),(1,5)=" +  
            k1.odleglosc(p2,0));  
        System.out.println(  
            "\nOdleglosc miedzy punktami\n(0,0),(2,10)=" +  
            k1.odleglosc());  
  
        System.out.println("\nPowierzchnia = "+ k1.pole());  
    }  
}
```

```
C:\Program Files\Xinox Software\JCreator...  
WspolrzednaX = 2  
WspolrzednaY = 10  
Dlugosc boku = 1  
  
WspolrzednaX = 4  
WspolrzednaY = 6  
  
Odleglosc miedzy punktami  
(4,6),(2,10)=4.47213595499958  
  
Suma odleglosci miedzy punktami  
(0,0),(2,10),(1,5)=15.297058540778353  
  
Odleglosc miedzy punktami  
(2,10),(1,5)=5.0990195135927845  
  
Odleglosc miedzy punktami  
(0,0),(2,10)=10.198039027185569  
  
Powierzchnia = 1  
Press any key to continue...
```


TABLICE (2), POLIMORFIZM

Przykład 3 - przechowanie w tablicy elementów obiektowych – **polimorfizm (przesłanianie metod)**

```
import java.lang.*;
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)); }
}
class Kwadrat extends Punkt
{ protected int dlugosc;
  public Kwadrat(int wspX, int wspY, int dlugosc_)
    { super(wspX,wspY);
      dlugosc=dlugosc_;}
  public double odleglosc()
    { return Math.sqrt(x*x+y*y); }
  public double odleglosc(Punkt p)
    { return odleglosc()+ super.odleglosc(p);}
  public int podajDl() { return dlugosc;} }
```

Operacje wynikające z polimorfizmu:

- Można przypisać do referencji klasy podstawowej referencje obiektów klas pochodnych
- Referencja **figury[i]** może zawierać referencję do obiektu typu **Punkt** lub **Kwadrat** – rozróżnia się te referencje i wywołuje metodę z właściwej klasy. Jest to możliwe dzięki przesłanianiu metod **odleglosc**

```
C:\Program Files\Xinox Software\JCreatorV3 LE\GE2...
0
1 ←
2
3
4
lancuch 0 ←
lancuch 1
lancuch 2
lancuch 3
lancuch 4
false, X=2, Y=2, odleglosc=0.0 ←
true, X=2, Y=2, odleglosc=2.8284271247461903
false, X=2, Y=2, odleglosc=0.0
true, X=2, Y=2, odleglosc=2.8284271247461903
Press any key to continue...
```

```
public class Tablice
{ public static void main(String args[])
{ final int N=5;
  int liczby [] = new int [N];
  for (int i=0; i<liczby.length; i++)
  { liczby[i]= i;
    System.out.println(liczby[i]); }
  String nazwy[]=new String[N];
  for (int i=0; i<nazwy.length; i++)
  { nazwy[i]=new String("lancuch "+i);
    System.out.println(nazwy[i]);}
  Punkt figury[]=new Punkt[N];
  for (int i=0; i<figury.length; i++)
  { figury[i]=new Punkt(2,2);
    if (i<figury.length-1)
      figury[++i]= new Kwadrat (2,2,2); }
  for (int i=0; i<figury.length-1;i++)
  { boolean p=figury[i] instanceof Kwadrat;
    System.out.println(p+
      ",X="+figury[i].podajX()+
      ", Y="+figury[i].podajY()+
      ", odleglosc="+figury[i].odleglosc(figury[i+1]));
  }
}
```

Przykład 4 - przechowanie w tablicy elementów obiektowych – polimorfizm (przesłanianie metod)

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

```
class Punkt
```

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

```
class Kwadrat extends Punkt
```

```
{ protected int dlugosc;
  public Kwadrat(int wspX, int wspY, int dl)
    { super(wspX,wspY);
      dlugosc=dl; }
  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=4;
  protected Punkt figury[]=new Punkt[N];
  public Figury()
    { for (int i=0; i<figury.length; i++)
      { figury[i]= new Punkt(i*20+5, i*20+5); //1
        if (i<figury.length-1)
          figury[++i]=
            new Kwadrat (i*30+10, i*30+10, i+30); //2
        } }
  public void rysuj(Graphics g)
    { for (int i=0; i<figury.length; i++)
      figury[i].rysuj(g); }
}
```

Operacje wynikające z polimorfizmu:

- Można przypisać do referencji klasy podstawowej referencje obiektów klas pochodnych (//1, //2)
- Referencja **figury[i]** może zawierać referencję do obiektu typu **Punkt** lub **Kwadrat** – rozróżnia się te referencje i wywołuje metodę **rysuj** z właściwej klasy dzięki przesłanianiu metod **rysuj**.

```

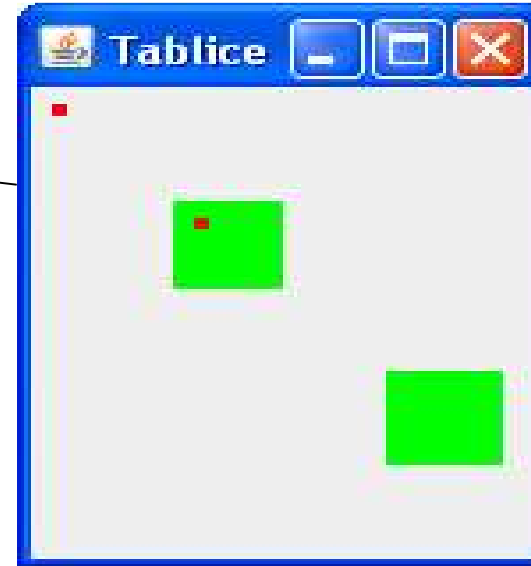
class Panel extends JPanel
{ Figury rys= new Figury();
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);
    rys.rysuj(g);
  }
}

```

```

public class Tablice extends JFrame
{ public Tablice()
  { super ("Tablice");
    setSize(150,200);
    setDefaultCloseOperation
      (JFrame.EXIT_ON_CLOSE);
    Panel obraz=new Panel();
    setContentPane(obraz);
  }
  public static void main(String args[])
  { Tablice tab=new Tablice();
    tab.setVisible(true);
  }
}

```



Obiekt klasy **JFrame** używa obiektu typu **JPanel**. Klasa **Tablice** dziedzicząca po klasie **JFrame** używa obiektu typu **Panel** dziedziczący po **JPanel**. Klasa **Panel** przesłoniła metodę **paintComponent**.

Używa ona wyspecjalizowanego obiektu typu **Figury** do rysowania figur na swojej powierzchni.

Każda klasa dziedzicząca po **JFrame** używa metody **paintComponent** wywoływanej od obiektu klasy z rodziny **JPanel** bez potrzeby rozróżniania typu tych obiektów, do rysowania treści graficznej ramki.

APLETY – ZASTOSOWANIE POLIMORFIZMU

Tryb graficzny: pakiety `java.awt` oraz `javax.swing`

- Aplety są uruchamiane w dowolnej przeglądarce sieciowej obsługującej język *Java* np. *NetScape Navigator*, *Microsoft Internet Explorer*
- Do uruchomienia apletu w celu przetestowania można korzystać z narzędzia *appletviewer* dołączonego do pakietu Javy
- Dołączenie dodatkowego modułu do przeglądarki sieciowej o nazwie *Java Plug-in* umożliwia korzystanie z rozszerzeń języka *Java 2* przy budowie apletu
- Należy utworzyć dwa pliki:
 - 1) Np. plik `MojAplet.java` z kodem źródłowym, czyli klasą która dziedziczy od klasy *Applet* i sprawia, że wykonany aplet staje się częścią przeglądarki

```

import java.awt.*;
import java.applet.*;
// przedefiniowanie metod klasy Applet

public class MojAplet extends Applet
{
    public void init()
        { // np. tworzenie obiektów ; tylko raz może być wykonana
        }
    public void start()
        { // ponowne uruchomienie apletu, można ją wielokrotnie wykonywać; wyświetlanie komunikatów,
          uruchamianie wątków }
    public void stop()
        { // wykonywana, gdy użytkownik opuszcza stronę lub gdy aplet sam wywoła
          // stop; np. zawieszanie wątku uruchomionego przez aplet
        }
    public void destroy()
        { // usuwanie danych roboczych, wątków itp. utworzonych przez aplet
        }
    public void paint(Graphics g)
        { // pozwala na wyświetlanie danych w swoim oknie dzięki obiektowi typu Graphics utworzonemu przez
          // przeglądarkę (import java.awt.Graphics;); reprezentuje on rysowany obszar w oknie
        }
}

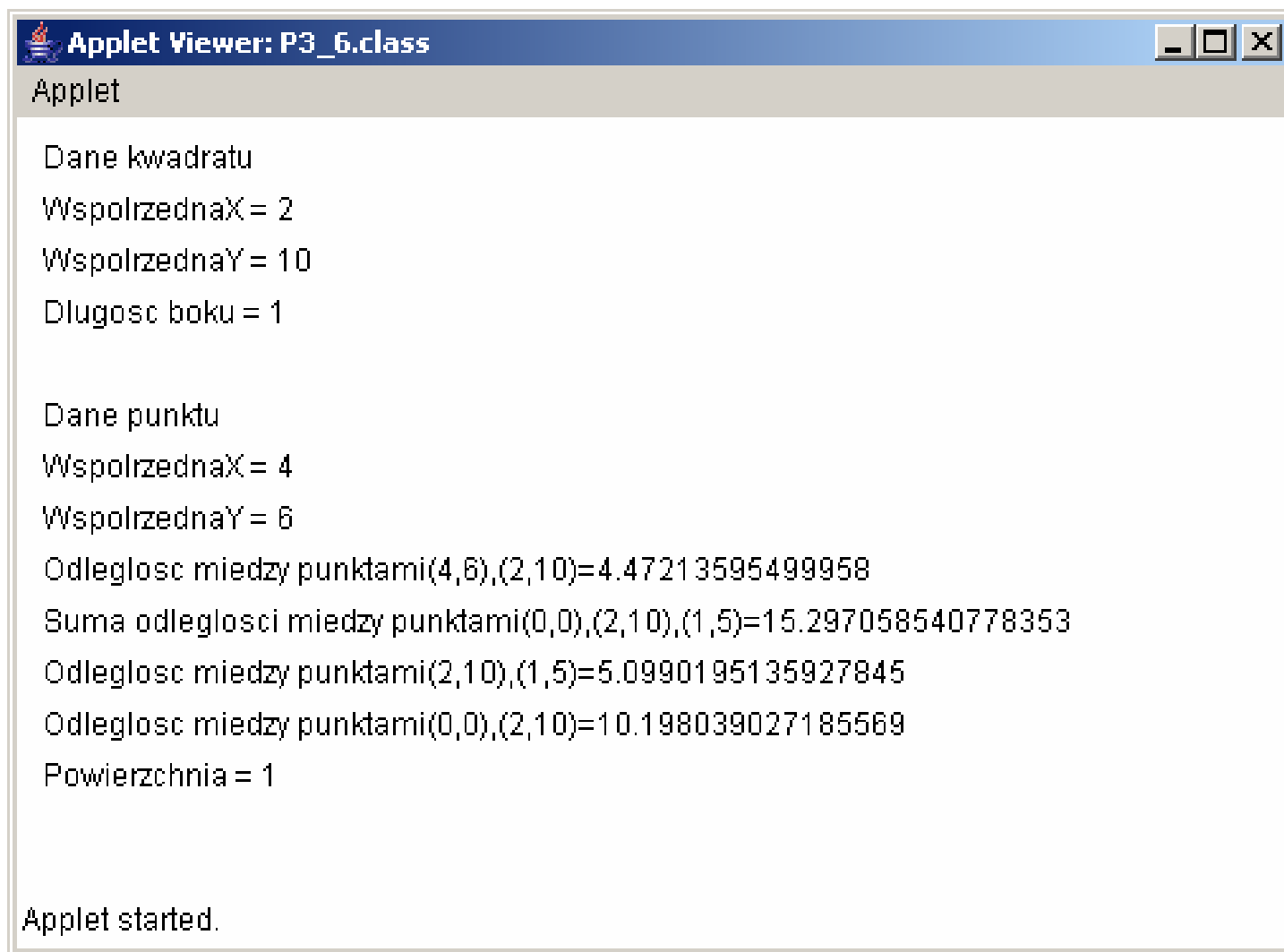
```

2) strona www zawierająca znacznik **<APPLET>**
(przykład 3)

3) wywołanie apletu wstawionego na stronę WWW

http://sprocket.ict.pwr.wroc.pl/~zkruczkiewicz/wyklady/java/classes/P3_6.html

Przykład 5 – Wykonanie apletu za pomocą programu *appletviewer.exe*. Aplet wykonuje w trybie graficznym operacje takie jak w programie z przykładu 1.



```
Applet Viewer: P3_6.class
Applet
Dane kwadratu
WspolrzednaX = 2
WspolrzednaY = 10
Dlugosc boku = 1

Dane punktu
WspolrzednaX = 4
WspolrzednaY = 6
Odleglosc miedzy punktami(4,6),(2,10)=4.47213595499958
Suma odleglosci miedzy punktami(0,0),(2,10),(1,5)=15.297058540778353
Odleglosc miedzy punktami(2,10),(1,5)=5.0990195135927845
Odleglosc miedzy punktami(0,0),(2,10)=10.198039027185569
Powierzchnia = 1

Applet started.
```

//plik P3_6.htm

```
<HTML>
<HEAD>
</HEAD>
<BODY BGCOLOR="000000">
<CENTER>
<APPLET
  code = "P3_6.class"
  width = "500"
  height = "300"
>
</APPLET>
</CENTER>
</BODY>
</HTML>
```



```
//zawartość pliku P3_6.java
```

```
import java.awt.*;
```

```
import java.applet.*;
```

```
//tutaj kod obu klas jak w przykładzie 1
```

```
public class P3_6 extends Applet
```

```
{
```

```
    public void paint(Graphics g)
```

```
    {
```

```
        g.drawString("Dane kwadratu", 10, 20);
```

```
        Kwadrat k1 = new Kwadrat(2,10,1);
```

```
        g.drawString("WspolrzednaX = "+k1.podajX(), 10, 40);
```

```
        g.drawString("WspolrzednaY = "+ k1.podajY(), 10, 60);
```

```
        g.drawString("Dlugosc boku = "+ k1.podajDl(), 10, 80);
```

```
        g.drawString("Dane punktu", 10, 120);
```

```
        Punkt p2 = new Punkt(4,6);
```

```
        g.drawString("WspolrzednaX = "+ p2.podajX(), 10, 140);
```

```
        g.drawString("WspolrzednaY = "+ p2.podajY(), 10, 160);
```

```
        g.drawString("Odleglosc miedzy punktami(4,6),(2,10)=" + p2.odleglosc(k1), 10, 180);
```

```
        p2.zmien(1,5);
```

```
        g.drawString("Suma odleglosci miedzy punktami(0,0),(2,10),(1,5)="+k1.odleglosc(p2), 10, 200);
```

```
        g.drawString("Odleglosc miedzy punktami(2,10),(1,5)=" + k1.odleglosc(p2,0), 10, 220);
```

```
        g.drawString("Odleglosc miedzy punktami(0,0),(2,10)="+ k1.odleglosc(), 10, 240);
```

```
        g.drawString("Powierzchnia = "+ k1.pole(), 10, 260);
```

```
    }
```

```
}
```

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

```
public class SwingApplet extends JApplet
```

```
{ String napis;
  int punkty=10;
```

```
public void init()
{ napis="init";
}
```

```
public void start()
{ napis+=" start";
}
```

```
public void stop()
{ napis+=" stop";
}
```

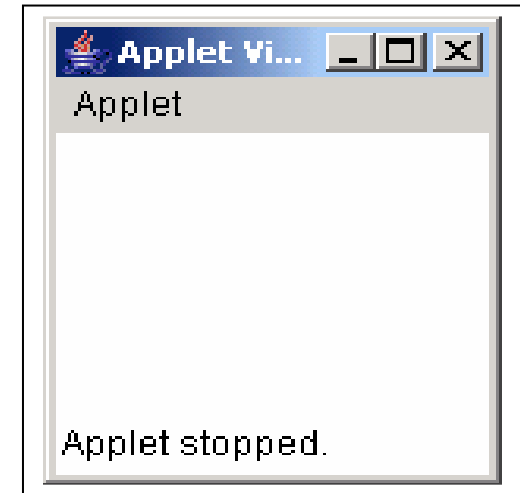
```
public void paint(Graphics g)
{ napis+=" paint | ";
  g.drawString(napis, 10, punkty);
}
```



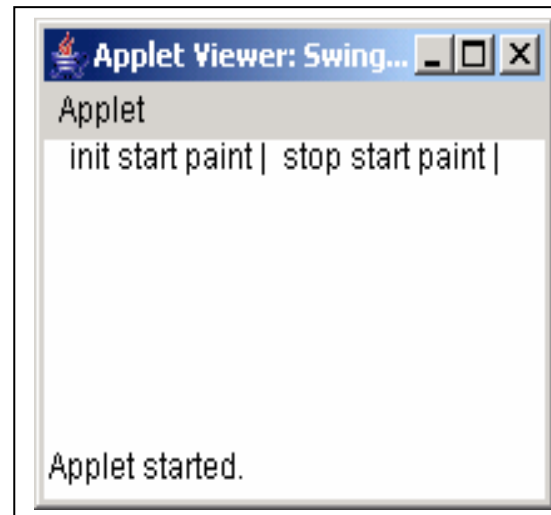
1)Uruchomiono aplet



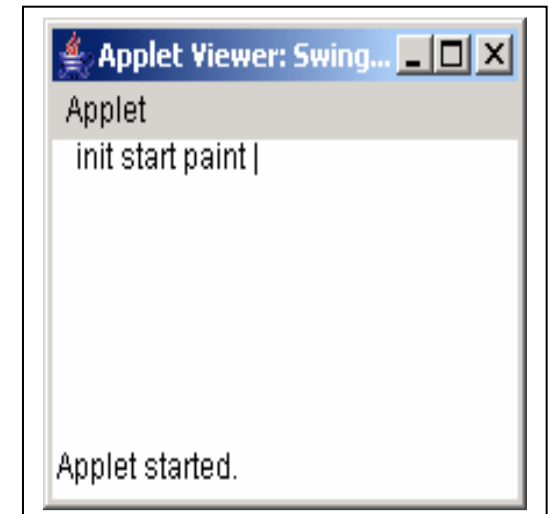
2)Zmieniono rozmiar okna



3) Wywołano: stop



4)Uruchomiono start aplet



5)Uruchomiono ponownie aplet

Przykład 6. –Aplet rysujący obiekty z rodziny klasy Punkt (1 sposób)

```
<html>
<head>
<title>Watch Applet</title>
</head>
<p> Figury:
<applet code="TabliceAp.class" height=250 width="345">
</applet>
</body>
</html>
```

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

```
//kody klas Punkt i Kwadrat nie zmieniły się
```

```
class Punkt
```

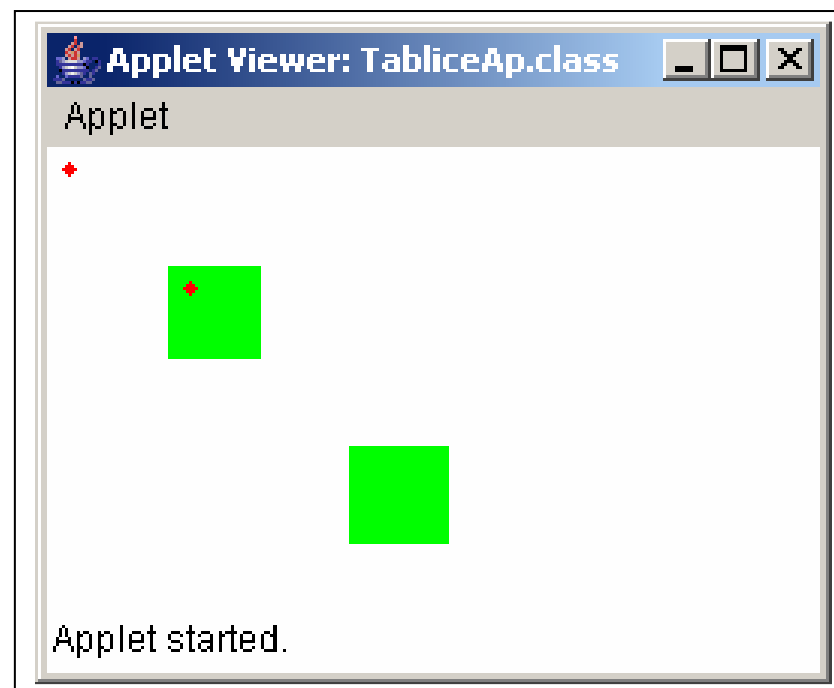
```
{int x, y;
```

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

```
void rysuj(Graphics g)
```

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

```
}
```



```

class Kwadrat extends Punkt
{
  int dlugosc;
  Kwadrat(int wspX, int wspY, int dlugosc_)
    {super(wspX,wspY);
      dlugosc=dlugosc_; }
  void rysuj(Graphics g)
  { Graphics2D g2D= (Graphics2D)g;
    Color pedzel = new Color(0,255,0);
    g2D.setColor(pedzel);
    g2D.fillRect(x,y,dlugosc,dlugosc); }
}

```

```

public class TabliceAp extends javax.swing.JApplet
{
  public void paint(Graphics g)
  { final int N=4;
    Punkt figury[]= new Punkt[N];
    for (int i=0; i<figury.length;i++)
    {
      figury[i]= new Punkt(i*20+5,i*20+5);
      if (i<figury.length-1)
        figury[++i]=new Kwadrat(i*30+10, i*30+10, i+30);}
    for (int i=0;i<figury.length; i++)
      figury[i].rysuj(g);
    repaint(); }
}

```

Przykład 7 - przechowanie w tablicy elementów obiektowych wg przykładu 4

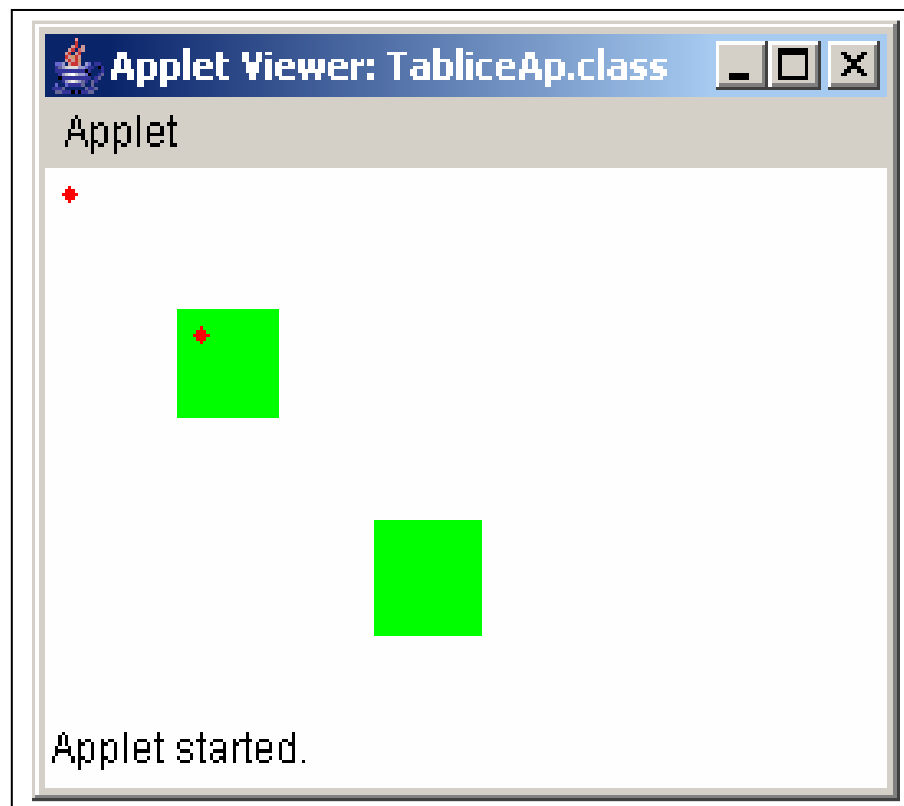
```
import java.lang.*;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;
```

```
class Punkt
```

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

```
class Kwadrat extends Punkt
```

```
{ protected int dlugosc;  
  public Kwadrat(int wspX, int wspY, int dl)  
    { super(wspX,wspY);  
      dlugosc=dl; }  
  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=4;
    protected Punkt figury[]=new Punkt[N];
    public Figury()
    {
        for (int i=0; i<figury.length; i++)
        {
            figury[i]= new Punkt(i*20+5, i*20+5); //1
            if (i<figury.length-1)
                figury[++i]= new Kwadrat (i*30+10, i*30+10, i+30); //2
        }
    }
    public void rysuj(Graphics g)
    {
        for (int i=0; i<figury.length; i++)
            figury[i].rysuj(g);
    }
}

```

```

class Panel extends JPanel
{
    Figury rys= new Figury();
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        rys.rysuj(g);
    }
}

```

```

public class TabliceAp extends javax.swing.JApplet
{
    public void init()
    {
        Panel obraz=new Panel();
        setContentPane(obraz);
    }
}

```