Sprawozdanie 5

Języki Programowania Obiektowego

Paweł Jamro

292510

1. Klasa SpringApplet

```
package com.company;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Timer;
import javax.swing.JPanel;

//implementujemy interfejs ActionListene

public class SpringApplet extends JApplet implements MouseMotionListener, MouseListener, ActionListener {

    private static final long serialVersionUID = 1L;
    private SimEngine simE;
    private SimTask simT;

    //pola do przechowywania obiektów powyższyoh klas

    private Timer czas;

    //pole do przechowywania wartośoi logicznej informujaca prawda czy nie o przeciagnieciu myszy
    private boolean MYSZ_STATE;
    private int x;
    private int y;

    //pola do przechowywanie elemenetow interfejsu
    private TextField mass, wspolczynnikK, wspolczynnikC, G, Lpoczatkowe;
    private Button reset;

    // metody do implementacji interfejsu
    @Override
    public void mouseMoved(MouseEvent arg0) {

    }
```

```
@Override
public void mouseClicked(MouseEvent e) {

}

@Override
public void mouseEntered(MouseEvent e) {

}

@Override
public void mouseExited(MouseEvent e){

}

@Override
public void mousePressed(MouseEvent e)
{

    //odczytujemy położenie myszy
    x = e.getX();
    y = e.getY();

    //sprawdzanie czy znajduje sie w obrebie mas
    if((x>=(int) simE.polozenieMasy.x-250 && x<=(int) simE.polozenieMasy.x+250)&&(y>=(int) simE.polozenieMasy.y-100  && y<=(int) simE.polozenieMasy.y +100))

    {   // wylaczenie timera
        czas.cancel();
        simE.reset();
        MYSZ_STATE = true;
    }

    e.consume(); // metoda comsume
}
```

```
public void mouseReleased(MouseEvent e)
{ // metda czy 3ywtapilo przeciagniecie
    if(MYSZ_STATE == true)
    {
        simT = new SimTask(simE, pole2: this,  odstep: 0.01);
        czas = new Timer();
        czas.scheduleAtFixedRate(simT,  delay: 0,  period: 1);
        MYSZ_STATE = false;
    }
    e.consume();
}

public void mouseDragged(MouseEvent arg0) {

    if (MYSZ_STATE == true) {
        simE.polozenieMasy.y = arg0.getY();
        repaint();
    }

    arg0.consume();
}
```

```
@Override
public void init()
{   // nowe obiekty
    this.setSize( width: 600, height: 600);

    simE = new SimEngine ( m1: 2, k1: 0.2, c1: 0.1, L1: 200, g1: 10, xMasy1: 100, yMasy1: 100, yPunkt1: 0);
    simT = new SimTask(simE, pole2: this, odstep: 0.01);
    czas = new Timer();
    czas.scheduleAtFixedRate(simT,  delay: 0,  period: 1);
    MYSZ_STATE = false;
    addMouseListener( l: this);
    addMouseListener( l: this);
    reset = new Button( label: "RESET");
    reset.addActionListener( l: this);
    add(reset);  // nadsluchiwanie i reset
    mass = new TextField( text: "1", columns: 4);
    wspolczynnikK=new TextField( text: "2", columns: 4);
    wspolczynnikC=new TextField( text: "3", columns: 4);
    Lpoczatkowe=new TextField( text: "4", columns: 4);
    G=new TextField( text: "5", columns: 4);
    // elementy GUi
    add(mass);
    add(wspolczynnikK);
    add(wspolczynnikC);
    add(Lpoczatkowe);
    add(G);
}
```

```java
@Override public void paint(Graphics gDC)
  //grafika
{   gDC.setColor(Color.ORANGE);
    gDC.fillRect( x: 0, y: 0, width: 600, height: 600);

    gDC.setColor(Color.black);
    gDC.drawLine( x1: (int) simE.showxPunkt() + 200, (int) simE.showyPunkt(),  x2: (int) simE.polozenieMasy.x + 200, (int) simE.polozenieMasy.y);



    gDC.setColor(Color.blue);
    gDC.fillOval(((int) simE.polozenieMasy.x + 100), (int) simE.polozenieMasy.y, width: 200,  height: 100);
  // przyciski
    mass.setBounds( x: 200, y: 500, width: 20, height: 20);
    wspolczynnikK.setBounds( x: 240, y: 500, width: 20, height: 20);
    wspolczynnikC.setBounds( x: 280, y: 500, width: 20, height: 20);
    Lpoczatkowe.setBounds( x: 320, y: 500, width: 20, height: 20);
    G.setBounds( x: 360, y: 500, width: 20, height: 20);
    reset.setBounds( x: 200, y: 540, width: 180, height: 40);
}

@Override
public void actionPerformed(ActionEvent e)
{    // metoda do implementacji interfesju ActionListener
    if(e.getSource() == reset)
    {
        czas.cancel();
        double masa=Double.parseDouble(mass.getText());
        double sprezytsc=Double.parseDouble(wspolczynnikK.getText());
        double tlumienie=Double.parseDouble(wspolczynnikC.getText());
        double l=Double.parseDouble(Lpoczatkowe.getText());
        double g=Double.parseDouble(G.getText());
        simE=new SimEngine(masa, sprezytsc, tlumienie,l,g,  xMasy1: 100, yMasy1: 20, yPunkt1: 0);
        simT=new SimTask(simE, pole2: this,  odstep: 0.01);
        czas=new Timer();
        czas.scheduleAtFixedRate(simT,  delay: 0,  period: 1);
        repaint();
    }
}
```

2. SimTask

```
package com.company;
import java.util.TimerTask;



public class SimTask extends TimerTask
{
private static final long serialVersionUID=1L;
private SimEngine poleklasy1; // przechowywanie obiektu klasy SimEngine
private SpringApplet poleklasy2; // przechowywanie obiekty klasy SpringApplet
private double czas; // przechowywanie odstepu

    // przypisanie klas
public SimTask(SimEngine pole1, SpringApplet pole2, double odstep)
    {
        this.poleklasy1=pole1;
        this.poleklasy2=pole2;
        this.czas=odstep;
    }

public void run()
    {
        poleklasy1.przebieg(czas);    //uruchomienie obliczenia kroku
        poleklasy2.repaint();
    }
}
```

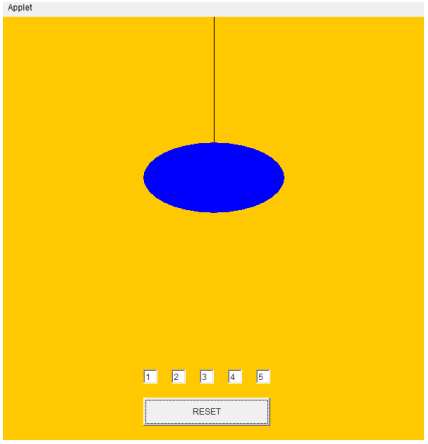
3. SimEngine

```java
package com.company;

public class SimEngine
{
    private double masa;   //masa
    private double sprezystosc;   //wspolczynnik sprezystosci
    private double tlumienie;   //wspolczynnik tlumienia
    private double dlugosc;  //dlugosc swobodna sprezyny
    private double xMasy; //polozenie masy
    private double yMasy;
    private double xPunkt;  //polozenie punktu zawieszenia
    private double yPunkt;
    private double skladowaVX; //predkosc
    private double skladowaVY;
    private double G; //przyspiesznie grawitacyjne
    public double ay;
    public double t;
    public double temp;

    Vector2D polozenieMasy; //wektor okreslajacy polozenie masy
    Vector2D polozeniePunkt;//wektor okreslajacy polozenie zawieszenia

    public void getM(double podajMase) //akcesory
    {
        masa = podajMase;
    }

    public void getSprezystosc(double podajSprezystosc) { sprezystosc = podajSprezystosc; }
    public void getTlumienie(double podajTlumienie) { tlumienie = podajTlumienie; }
    public void getdlugosc(double podajdlugosc) {dlugosc = podajdlugosc; }
    public void getxMasy(double podajxMasy) { xMasy = podajxMasy; }
    public void getyMasy(double podajyMasy) { yMasy = podajyMasy; }
    public void getxPunkt(double podajxPunkt) { xPunkt = podajxPunkt; }
    public void getyPunkt(double podajyPunkt) { yPunkt = podajyPunkt; }
    public void getyV(double podajyV) { skladowaVY = podajyV; }
    public void getG(double podajG) { G = podajG; }
    public double showxPunkt() { return polozenieMasy.x; }
    public double showyPunkt() { return polozeniePunkt.y; }
    public double showxMasy() { return polozenieMasy.x; }

    public double showyMasy() {
        System.out.println(polozenieMasy.y);
        return polozenieMasy.y;
    }

    public SimEngine(double m1, double k1, double c1, double L1, double g1, double xMasy1, double yMasy1, double yPunkt1) {
        masa= m1;
        sprezystosc = k1;
        tlumienie = c1;
        dlugosc = L1;
        skladowaVX = 0;
        skladowaVY = 0;
        G = g1;
        ay = 0;
        xMasy = xMasy1;
        yMasy = yMasy1;
        xPunkt = xMasy1;  // zawieszenie ich w jednej linii
        yPunkt = yPunkt1;
        t = 0;
        polozenieMasy = new Vector2D(xMasy1, yMasy1);
        polozeniePunkt = new Vector2D(xMasy1, yPunkt1);
    }

    // metoda umozliwiajaca obliczenie przebiegu symulacji

    public void przebieg(double krok) {
        t = krok;
        ay = (masa* G - sprezystosc * polozenieMasy.y - tlumienie * skladowaVY) / masa;
        skladowaVY = skladowaVY + ay * t;
        temp = temp + skladowaVY * t + ay * t * t / 2;
        polozenieMasy.y = (int) temp;
    }

    public void reset() { skladowaVY = 0; }
}
```

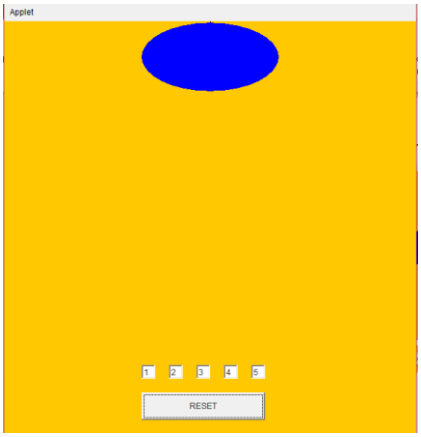
4. Vector2D

```java
package com.company;

public class Vector2D {
    public double x; // współrzędne wektora, publiczne
    public double y;

    public Vector2D() {
        //Konstruktor domyślny
        x = 0;
        y = 0;
    }
    //możemy nadać dowolne wartości polom klasy, konstruktor domyślny

    public Vector2D(double x1, double y1) {
        x = x1;
        y = y1;
    }

}
```

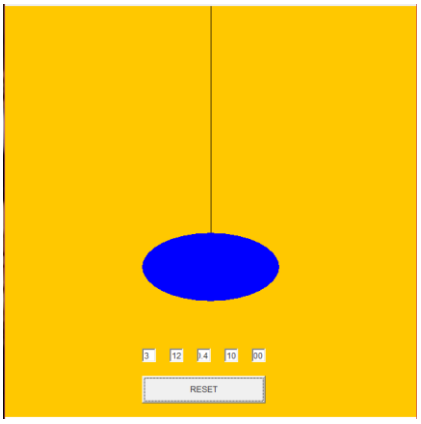
5. Działanie programu:



Z wartościami początkowymi



Po zresetowaniu, położenie 0.



Po zmianie ustawień.

6. Odpowiedź na pytanie:

Interfejs- są definicją typów, tyle że nieco uboższą niż klasy. Interfejsy mogą zawierać tylko stałe i deklaracje metod- bez implementacji.

```
{modyfikatory interfejsu} interface {nazwa interfejsu}
{
  {
   Deklaracje stałych i metod
  }
{
```