Sprawozdanie 4

Języki Programowania Obiektowego

Paweł Jamro

292510

Inżynieria Mechatroniczna

1. Klasa SpringApplet

```
package com.company;
import ...


public class SpringApplet extends JApplet{
    private static final long serialVersionUID = 1L;
    private SimEngine simE;
    private SimTask simT;
    private Timer czas;

    // bezparametrowa, publiczna metoda
    @Override public void init() {
        //tworzenie nowych obiektów klas
        this.setSize( width: 400, height: 300);
        simE=new SimEngine( m1: 4, k1: 0.5, c1: 0.04, L1: 399, g1: 10 , xMasy1: 200, yMasy1: 200 , yPunkt1: 0); //prywatne pola do przechowywania obiektów
        simT=new SimTask(simE, pole2: this, odstep: 0.01);                                                      // poniższych klas
        czas=new Timer();
        czas.scheduleAtFixedRate(simT, delay: 0, period: 1);
    }

    @Override public void paint(Graphics gDC){

        //czesc graficzna, rysowanie poszczegolnch elementow

        gDC.setColor(Color.ORANGE);
        gDC.fillRect( x: 0, y: 0, width: 400, height: 300);

        gDC.setColor(Color.black);
        gDC.drawLine((int) simE.showxPunkt(),(int) simE.showyPunkt(), (int) simE.showxMasy(), (int) simE.showyMasy());

        gDC.setColor(Color.blue);
        gDC.fillRect(((int) simE.showxMasy()-75), (int) simE.showyMasy(), width: 150, height: 75);

        // parametry masy zawieszonej na nici
    }
}
```

2. Klasa 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 swodobna 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; }
}
```

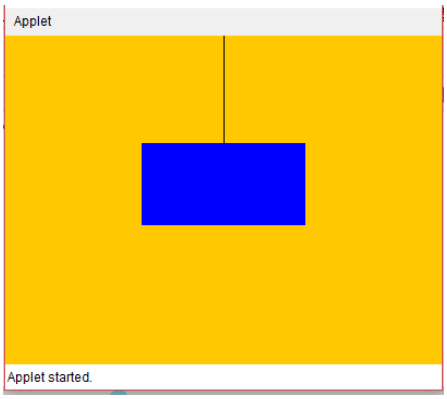
3. Klasa SimTask

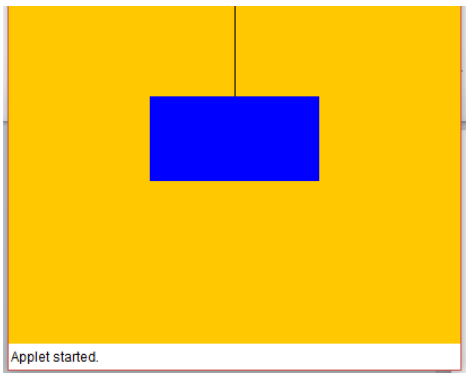
```
package com.company;
import java.util.TimerTask;



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

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();
    }
}
```

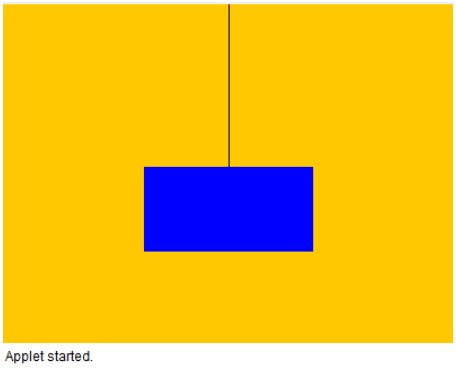
4. Vector2D

```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. Wynik działania programu testowego + krótkie omówienie







Zamieszczone zdjęcia przedstawiają masę zawieszoną na sprężynie (przedstawiona jest w aplecie jako czarna linia). Wprawiona w drgania oscyluje góra - dół, a potem zgodnie z prawami fizyki zaczyna dążyć do położenia równowagi.

6. Odpowiedź na pytania

1. Czym są metody abstrakcyjne?

Metody abstrakcyjne są to takie metody, które nie posiadają implementacji (ani nawet nawiasów klamrowych). Zawarte są w klasach abstrakcyjnych, które to dzięki takim metodom mogą rozbudować swoją funkcjonalność.

2. Czym są pola statyczne i metody statyczne?

Pole danych lub metoda może zadeklarowana z modyfikatorem static. Taka deklaracja oznacza, że pole danych lub metoda dotyczy klasy a nie obiektu, tzn. dla wszystkich obiektów danej klasy pole statyczne ma tę samą wartość. Odwołanie do statycznego pola danych może mieć postać: NazwaKlasy.PoleDanych , a dla metod statycznych NazwaKlasy.Metoda(). Metody statyczne podobnie jak statyczne pola danych są przypisane do klasy a nie konkretnego obiektu i służą do operacji tylko na polach statycznych.