**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

КУРСОВА РАБОТА  
ПО ПРОГРАМНИ ЕЗИЦИ

Студент:

Факултетен номер:

Група:

Специалност:

Дата: Изготвил:  
 Проверил:

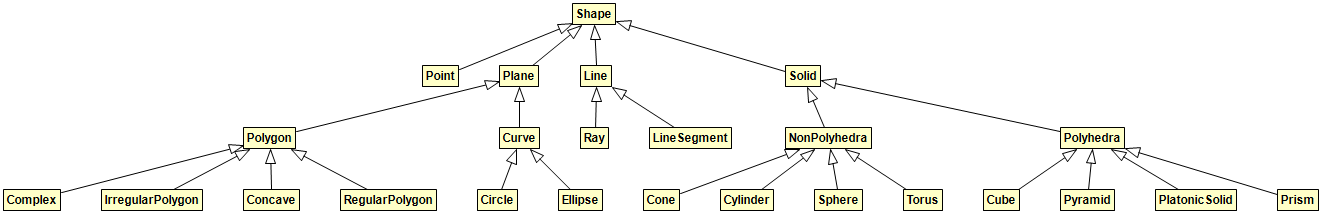
# Задание.

Да се състави йерархия от класове, описващи **геометрични фигури** и програма на Java с графичен потреителски интерфейс, включваща следните функционалности:

* въвеждане на данни за обекти от тези класове;
* съхранение на данните;

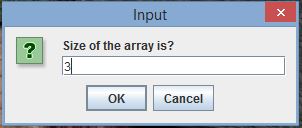
Данните за обектите да се попълват от **графичен интерфейс, като след избора на конкретен обект, атрибутите му се въвеждат в отделен панел в същия прозорец (може например да се използва CardLayout и отделните панели за всички класове да са един върху друг)** и да се съхраняват в **масив**. Изборът на атрибути и методи е на студента, но броят, типовете им и разположението име в йерархията влияят на оценката.

# Клас диаграми.

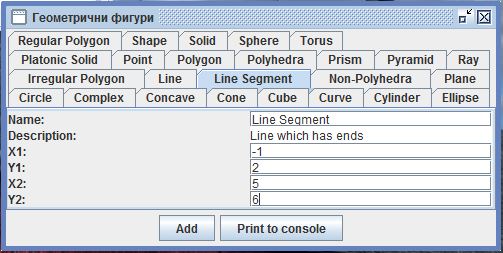


Фигура 1 Клас диаграма.

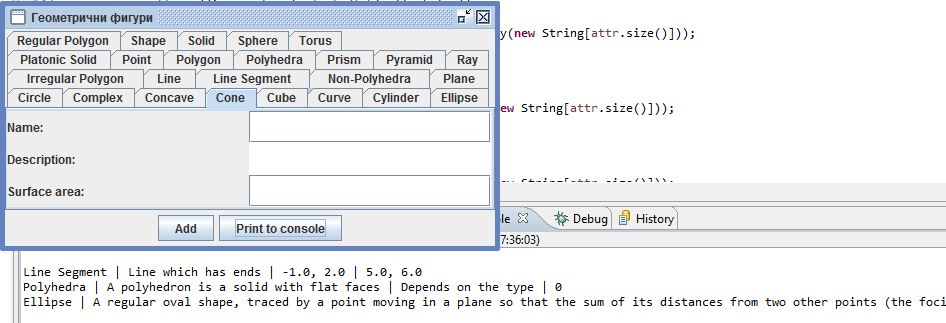
# Тестови примери



Фигура 2 Въвеждане размера на масива



Фигура 3 Добавяне на данни



Фигура 4 Печатане на данни на конзолата

# Листинг.

MAIN.JAVA

package geofig;

public class Main {

private static GUI gui;

public static void main(String[] args) {

setGui(new GUI("Геометрични фигури"));

}

public static GUI getGui() {

return gui;

}

public static void setGui(GUI gui) {

Main.gui = gui;

}

}

GUI.JAVA

package geofig;

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import geofig.shapes.\*;

import geofig.shapes.forms.\*;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTabbedPane;

public class GUI {

// прозорецът на програмата

private JFrame frame;

// панелът с табовете - за избор на клас

private JTabbedPane tabbedPane;

// брояч на добавените елементи в масива

private int indexCounter = 0;

// размер на масива

private int length;

// масивът с обектите

private Shape[] shapes;

// конструктор

public GUI(String title) {

try {

// показване на диалог за въвеждане на размера на масива

// т.к диалогът връща String - конвертираме го в число

length = Integer.parseUnsignedInt(JOptionPane.showInputDialog(frame, "Size of the array is?"));

}

catch(Exception ex) {

// ако потребителят не е въвел число или въобще не е въвел нищо

// показване на диалог със съответната информация

JOptionPane.showMessageDialog(frame, "Invalid number!");

}

// ако числото, което е въвел е по-голям от 0

if (length > 0) {

// създаване на масива

shapes = new Shape[length];

// създаване и добавяне на нужните панели в tabbedPane

initTabbedPane();

// прозорецът да ползва ресурсите на Java за визуализация

JFrame.setDefaultLookAndFeelDecorated(true);

// създаване на прозореца

frame = new JFrame(title);

// при избор на Х в горния-десен ъгъл да се спира приложението

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// задаване типа на подреждане на елементите в прозореца - BorderLayout

frame.setLayout(new BorderLayout());

// добавяне на tabbedPane в центъра на прозореца

frame.add(tabbedPane, BorderLayout.CENTER);

// създаване на панел, който ще съдържа двата бутона:

// - добавяне на обекта в масива

// - печатане на масива в конзолата

JPanel panel = new JPanel();

// създаване на бутона за добавяне

JButton btnAdd = new JButton("Add");

// добавяне на слушател

btnAdd.addActionListener(new ActionListener() {

// изпълнява се само при кликване на бутона

@Override

public void actionPerformed(ActionEvent e) {

try {

// ако има място в масива

if (indexCounter < shapes.length) {

// добавяне на обекта в масива

boolean isAdded = addItem(tabbedPane.getSelectedComponent(),

tabbedPane.getTitleAt(tabbedPane.getSelectedIndex()));

// ако НЕ е добавен успешно

if (!isAdded) {

// извеждане на съобщение

JOptionPane.showMessageDialog(frame, "Please fill out all fields!");

}

else {

JOptionPane.showMessageDialog(frame, "Added!");

}

}

else {

// ако няма място в масива - извеждане на съобщение

JOptionPane.showMessageDialog(frame, "Array is full!");

}

}

catch(Exception ex) {

// ако вместо число се въведе текст се извежда съобщение за грешка

JOptionPane.showMessageDialog(frame, "Please enter correct information!");

}

}

});

// добавяне на бутона в панела

panel.add(btnAdd);

// създаване на бутона за печатане

JButton btnPrint = new JButton("Print to console");

// добавяне на слушател

btnPrint.addActionListener(new ActionListener() {

// изпълнява се само при кликване на бутона

@Override

public void actionPerformed(ActionEvent e) {

// с цикъл for обхождаме масива и извеждаме обектите в него

for (Shape s : shapes) {

if (s != null) {

System.out.print(s.toString());

}

}

}

});

// добавяне на бутона в панела

panel.add(btnPrint);

// добавяне на панела в прозореца

frame.add(panel, BorderLayout.SOUTH);

// пакетиране на компонентите на прозореца

frame.pack();

// задаване размера на прозореца

frame.setSize(500, 250);

// да не може да се оразмерява - прозорецът

frame.setResizable(false);

// да се появява в центъра на екрана на монитора

frame.setLocationRelativeTo(null);

// прозорецът да е видим

frame.setVisible(true);

}

else {

JOptionPane.showMessageDialog(frame, "Please enter a number bigger than zero!");

}

}

private void initTabbedPane() {

// създаване на панела с табовете

tabbedPane = new JTabbedPane();

// добавяне на панелите

tabbedPane.add(Shapes.CIRCLE, new CircleForm());

tabbedPane.add(Shapes.COMPLEX, new ComplexForm());

tabbedPane.add(Shapes.CONCAVE, new ConcaveForm());

tabbedPane.add(Shapes.CONE, new ConeForm());

tabbedPane.add(Shapes.CUBE, new CubeForm());

tabbedPane.add(Shapes.CURVE, new CurveForm());

tabbedPane.add(Shapes.CYLINDER, new CylinderForm());

tabbedPane.add(Shapes.ELLIPSE, new EllipseForm());

tabbedPane.add(Shapes.IRREGULAR\_POLYGON, new IrregularPolygonForm());

tabbedPane.add(Shapes.LINE, new LineForm());

tabbedPane.add(Shapes.LINE\_SEGMENT, new LineSegmentForm());

tabbedPane.add(Shapes.NON\_POLYHEDRA, new NonPolyhedraForm());

tabbedPane.add(Shapes.PLANE, new PlaneForm());

tabbedPane.add(Shapes.PLATONIC\_SOLID, new PlatonicSolidForm());

tabbedPane.add(Shapes.POINT, new PointForm());

tabbedPane.add(Shapes.POLYGON, new PolygonForm());

tabbedPane.add(Shapes.POLYHEDRA, new PolyhedraForm());

tabbedPane.add(Shapes.PRISM, new PrismForm());

tabbedPane.add(Shapes.PYRAMID, new PyramidForm());

tabbedPane.add(Shapes.RAY, new RayForm());

tabbedPane.add(Shapes.REGULAR\_POLYGON, new RegularPolygonForm());

tabbedPane.add(Shapes.SHAPE, new ShapeForm());

tabbedPane.add(Shapes.SOLID, new SolidForm());

tabbedPane.add(Shapes.SPHERE, new SphereForm());

tabbedPane.add(Shapes.TORUS, new TorusForm());

}

private boolean addItem(Component selectedTab, String title) {

ArrayList<String> attr = null;

int oldIndex = indexCounter;

// намиране на избрания клас, извличане на данните, въведени

// в панела и предаване на тези данни в конструктора на обекта за добавяне.

// Съответният клас се грижи за тяхното конвертиране в нужния тип от String.

// Ако има празно поле - обектът не се добавя в масива

switch (title) {

case Shapes.CIRCLE:

attr = ((CircleForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Circle(attr.toArray(new String[attr.size()]));

break;

case Shapes.COMPLEX:

attr = ((ComplexForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Complex(attr.toArray(new String[attr.size()]));

break;

case Shapes.CONCAVE:

attr = ((ConcaveForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Concave(attr.toArray(new String[attr.size()]));

break;

case Shapes.CONE:

attr = ((ConeForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Cone(attr.toArray(new String[attr.size()]));

break;

case Shapes.CUBE:

attr = ((CubeForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Cube(attr.toArray(new String[attr.size()]));

break;

case Shapes.CURVE:

attr = ((CurveForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Curve(attr.toArray(new String[attr.size()]));

break;

case Shapes.CYLINDER:

attr = ((CylinderForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Cylinder(attr.toArray(new String[attr.size()]));

break;

case Shapes.ELLIPSE:

attr = ((EllipseForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Ellipse(attr.toArray(new String[attr.size()]));

break;

case Shapes.IRREGULAR\_POLYGON:

attr = ((IrregularPolygonForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new IrregularPolygon(attr.toArray(new String[attr.size()]));

break;

case Shapes.LINE:

attr = ((LineForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Line(attr.toArray(new String[attr.size()]));

break;

case Shapes.LINE\_SEGMENT:

attr = ((LineSegmentForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new LineSegment(attr.toArray(new String[attr.size()]));

break;

case Shapes.NON\_POLYHEDRA:

attr = ((NonPolyhedraForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new NonPolyhedra(attr.toArray(new String[attr.size()]));

break;

case Shapes.PLANE:

attr = ((PlaneForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Plane(attr.toArray(new String[attr.size()]));

break;

case Shapes.PLATONIC\_SOLID:

attr = ((PlatonicSolidForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new PlatonicSolid(attr.toArray(new String[attr.size()]));

break;

case Shapes.POINT:

attr = ((PointForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Point(attr.toArray(new String[attr.size()]));

break;

case Shapes.POLYGON:

attr = ((PolygonForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Polygon(attr.toArray(new String[attr.size()]));

break;

case Shapes.POLYHEDRA:

attr = ((PolyhedraForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Polyhedra(attr.toArray(new String[attr.size()]));

break;

case Shapes.PRISM:

attr = ((PrismForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Prism(attr.toArray(new String[attr.size()]));

break;

case Shapes.PYRAMID:

attr = ((PyramidForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Pyramid(attr.toArray(new String[attr.size()]));

break;

case Shapes.RAY:

attr = ((RayForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Ray(attr.toArray(new String[attr.size()]));

break;

case Shapes.REGULAR\_POLYGON:

attr = ((RegularPolygonForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new RegularPolygon(attr.toArray(new String[attr.size()]));

break;

case Shapes.SHAPE:

attr = ((ShapeForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Shape(attr.toArray(new String[attr.size()]));

break;

case Shapes.SOLID:

attr = ((SolidForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Solid(attr.toArray(new String[attr.size()]));

break;

case Shapes.SPHERE:

attr = ((SphereForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Sphere(attr.toArray(new String[attr.size()]));

break;

case Shapes.TORUS:

attr = ((TorusForm)selectedTab).attributes();

if (!containsEmptyValue(attr))

shapes[indexCounter] = new Torus(attr.toArray(new String[attr.size()]));

break;

default:

System.out.println("\"" + title + "\" is invalid name!");

break;

}

if (shapes[oldIndex] == null) {

return false;

}

indexCounter++;

return true;

}

// фунцкия за проверка дали има празен елемент в списъка list

private boolean containsEmptyValue(ArrayList<String> list) {

for (String item : list) {

if (null == item || item.isEmpty()) {

return true;

}

}

return false;

}

}

CIRCLE.JAVA

package geofig.shapes;

public class Circle extends Curve {

private float circumference;

public Circle() {

}

public Circle(String name, String description, String area,

float majorAxes, float minorAxes, float circumference) {

super(name, description, area, majorAxes, minorAxes);

this.circumference = circumference;

}

public Circle(String[] attributes) {

this(attributes[0], attributes[1], attributes[2],

Float.parseFloat(attributes[3]),

Float.parseFloat(attributes[4]),

Float.parseFloat(attributes[5]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(circumference);

return builder.toString();

}

public float getCircumference() {

return circumference;

}

public void setCircumference(float circumference) {

this.circumference = circumference;

}

}

CIRCLEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class CircleForm extends CurveForm {

private static final long serialVersionUID = 1L;

private JTextField circumference;

public CircleForm() {

super();

this.add(new JLabel("Circumference:"));

this.circumference = new JTextField();

this.add(this.circumference);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(circumference.getText());

return result;

}

}

COMPLEX.JAVA

package geofig.shapes;

public class Complex extends Polygon {

public Complex() {

}

public Complex(String name, String description, String area,

int sides) {

super(name, description, area, sides);

}

public Complex(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

COMPLEXFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class ComplexForm extends PolygonForm {

private static final long serialVersionUID = 1L;

public ComplexForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

CONCAVE.JAVA

package geofig.shapes;

public class Concave extends Polygon {

public Concave() {

}

public Concave(String name, String description, String area,

int sides) {

super(name, description, area, sides);

}

public Concave(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

CONCAVEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class ConcaveForm extends PolygonForm {

private static final long serialVersionUID = 1L;

public ConcaveForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

CONE.JAVA

package geofig.shapes;

public class Cone extends NonPolyhedra {

public Cone() {

}

public Cone(String name, String description, String surfaceArea) {

super(name, description, surfaceArea);

}

public Cone(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

CONEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class ConeForm extends NonPolyhedraForm {

private static final long serialVersionUID = 1L;

public ConeForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

CUBE.JAVA

package geofig.shapes;

public class Cube extends Polyhedra {

public Cube() {

}

public Cube(String name, String description, String surfaceArea,

int vertices) {

super(name, description, surfaceArea, vertices);

}

public Cube(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

CUBEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class CubeForm extends PolyhedraForm {

private static final long serialVersionUID = 1L;

public CubeForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

CURVE.JAVA

package geofig.shapes;

public class Curve extends Plane {

private float majorAxes;

private float minorAxes;

public Curve() {

}

public Curve(String name, String description, String area,

float majorAxes, float minorAxes) {

super(name, description, area);

this.majorAxes = majorAxes;

this.minorAxes = minorAxes;

}

public Curve(String[] attributes) {

this(attributes[0], attributes[1],

attributes[2],

Float.parseFloat(attributes[3]),

Float.parseFloat(attributes[4]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(majorAxes).append(" | ");

builder.append(minorAxes);

return builder.toString();

}

public float getMajorAxes() {

return majorAxes;

}

public void setMajorAxes(float majorAxes) {

this.majorAxes = majorAxes;

}

public float getMinorAxes() {

return minorAxes;

}

public void setMinorAxes(float minorAxes) {

this.minorAxes = minorAxes;

}

}

CURVEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class CurveForm extends PlaneForm {

private static final long serialVersionUID = 1L;

private JTextField majorAxes;

private JTextField minorAxes;

public CurveForm() {

super();

this.add(new JLabel("Major axes:"));

this.majorAxes = new JTextField();

this.add(this.majorAxes);

this.add(new JLabel("Minor axes:"));

this.minorAxes = new JTextField();

this.add(this.minorAxes);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(majorAxes.getText());

result.add(minorAxes.getText());

return result;

}

}

CYLINDER.JAVA

package geofig.shapes;

public class Cylinder extends NonPolyhedra {

public Cylinder() {

}

public Cylinder(String name, String description, String surfaceArea) {

super(name, description, surfaceArea);

}

public Cylinder(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

CYLINDERFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class CylinderForm extends NonPolyhedraForm {

private static final long serialVersionUID = 1L;

public CylinderForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

ELLIPSE.JAVA

package geofig.shapes;

public class Ellipse extends Curve {

private String specialCase;

public Ellipse() {

}

public Ellipse(String name, String description, String area,

float majorAxes, float minorAxes) {

super(name, description, area, majorAxes, minorAxes);

}

public Ellipse(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(this.getSpecialCase());

return builder.toString();

}

public String getSpecialCase() {

if (this.getMajorAxes() == this.getMinorAxes()) {

specialCase = "Circle: major axes = minor axes.";

}

else {

specialCase = "Ellipse: major axes != minor axes.";

}

return specialCase;

}

public void setSpecialCase(String specialCase) {

this.specialCase = specialCase;

}

}

ELLIPSEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class EllipseForm extends CurveForm {

private static final long serialVersionUID = 1L;

public EllipseForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

IRREGULARPOLYGON.JAVA

package geofig.shapes;

public class IrregularPolygon extends Polygon {

public IrregularPolygon() {

}

public IrregularPolygon(String name, String description, String area,

int sides) {

super(name, description, area, sides);

}

public IrregularPolygon(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

IRREGULARPOLYGONFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class IrregularPolygonForm extends PolygonForm {

private static final long serialVersionUID = 1L;

public IrregularPolygonForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

LINE.JAVA

package geofig.shapes;

public class Line extends Shape {

public Line() {

}

public Line(String name, String description) {

super(name, description);

}

public Line(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

LINEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class LineForm extends ShapeForm {

private static final long serialVersionUID = 1L;

public LineForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

LINESEGMENT.JAVA

package geofig.shapes;

public class LineSegment extends Line {

private float x1;

private float y1;

private float x2;

private float y2;

public LineSegment() {

}

public LineSegment(String name, String description,

float x1, float y1, float x2, float y2) {

super(name, description);

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

}

public LineSegment(String[] attributes) {

this(attributes[0], attributes[1],

Float.parseFloat(attributes[2]),

Float.parseFloat(attributes[3]),

Float.parseFloat(attributes[4]),

Float.parseFloat(attributes[5]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(x1).append(", ");

builder.append(y1).append(" | ");

builder.append(x2).append(", ");

builder.append(y2);

return builder.toString();

}

public float getX1() {

return x1;

}

public void setX1(float x1) {

this.x1 = x1;

}

public float getY1() {

return y1;

}

public void setY1(float y1) {

this.y1 = y1;

}

public float getX2() {

return x2;

}

public void setX2(float x2) {

this.x2 = x2;

}

public float getY2() {

return y2;

}

public void setY2(float y2) {

this.y2 = y2;

}

}

LINESEGMENTFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class LineSegmentForm extends LineForm {

private static final long serialVersionUID = 1L;

private JTextField x1;

private JTextField y1;

private JTextField x2;

private JTextField y2;

public LineSegmentForm() {

super();

this.add(new JLabel("X1:"));

this.x1 = new JTextField();

this.add(this.x1);

this.add(new JLabel("Y1:"));

this.y1 = new JTextField();

this.add(this.y1);

this.add(new JLabel("X2:"));

this.x2 = new JTextField();

this.add(this.x2);

this.add(new JLabel("Y2:"));

this.y2 = new JTextField();

this.add(this.y2);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(x1.getText());

result.add(y1.getText());

result.add(x2.getText());

result.add(y2.getText());

return result;

}

}

NONPOLYHEDRA.JAVA

package geofig.shapes;

public class NonPolyhedra extends Solid {

public NonPolyhedra() {

}

public NonPolyhedra(String name, String description, String surfaceArea) {

super(name, description, surfaceArea);

}

public NonPolyhedra(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

NONPOLYHEDRAFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class NonPolyhedraForm extends SolidForm {

private static final long serialVersionUID = 1L;

public NonPolyhedraForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

PLANE.JAVA

package geofig.shapes;

public class Plane extends Shape {

private String area;

public Plane() {

}

public Plane(String name, String description, String area) {

super(name, description);

this.area = area;

}

public Plane(String[] attributes) {

this(attributes[0], attributes[1], attributes[2]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(area);

return builder.toString();

}

public String getArea() {

return area;

}

public void setArea(String area) {

this.area = area;

}

}

PLANEFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PlaneForm extends ShapeForm {

private static final long serialVersionUID = 1L;

private JTextField area;

public PlaneForm() {

super();

this.add(new JLabel("Area:"));

this.area = new JTextField();

this.add(this.area);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(area.getText());

return result;

}

}

PLATONICSOLID.JAVA

package geofig.shapes;

public class PlatonicSolid extends Polyhedra {

public PlatonicSolid() {

}

public PlatonicSolid(String name, String description, String surfaceArea,

int vertices) {

super(name, description, surfaceArea, vertices);

}

public PlatonicSolid(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

PLATONICSOLIDFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class PlatonicSolidForm extends PolyhedraForm {

private static final long serialVersionUID = 1L;

public PlatonicSolidForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

POINT.JAVA

package geofig.shapes;

public class Point extends Shape {

private float abscissa;

private float ordinate;

public Point() {

}

public Point(String name, String description, float abscissa, float ordinate) {

super(name, description);

this.abscissa = abscissa;

this.ordinate = ordinate;

}

public Point(String[] attributes) {

this(attributes[0], attributes[1],

Float.parseFloat(attributes[2]),

Float.parseFloat(attributes[3]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(abscissa).append(", ");

builder.append(ordinate);

return builder.toString();

}

public float getAbscissa() {

return abscissa;

}

public void setAbscissa(float abscissa) {

this.abscissa = abscissa;

}

public float getOrdinate() {

return ordinate;

}

public void setOrdinate(float ordinate) {

this.ordinate = ordinate;

}

}

POINTFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PointForm extends ShapeForm {

private static final long serialVersionUID = 1L;

private JTextField abscissa;

private JTextField ordinate;

public PointForm() {

super();

this.add(new JLabel("Abscissa:"));

this.abscissa = new JTextField();

this.add(this.abscissa);

this.add(new JLabel("Ordinate:"));

this.ordinate = new JTextField();

this.add(this.ordinate);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(abscissa.getText());

result.add(ordinate.getText());

return result;

}

}

POLYGON.JAVA

package geofig.shapes;

public class Polygon extends Plane {

private int sides;

public Polygon() {

// TODO Auto-generated constructor stub

}

public Polygon(String name, String description, String area, int sides) {

super(name, description, area);

this.sides = sides;

}

public Polygon(String[] attributes) {

this(attributes[0], attributes[1],

attributes[2], Integer.parseInt(attributes[3]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(sides);

return builder.toString();

}

public int getSides() {

return sides;

}

public void setSides(int sides) {

this.sides = sides;

}

}

POLYGONFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PolygonForm extends PlaneForm {

private static final long serialVersionUID = 1L;

private JTextField sides;

public PolygonForm() {

super();

this.add(new JLabel("Sides:"));

this.sides = new JTextField();

this.add(this.sides);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(sides.getText());

return result;

}

}

POLYHEDRA.JAVA

package geofig.shapes;

public class Polyhedra extends Solid {

private int vertices;

public Polyhedra() {

}

public Polyhedra(String name, String description, String surfaceArea,

int vertices) {

super(name, description, surfaceArea);

this.vertices = vertices;

}

public Polyhedra(String[] attributes) {

this(attributes[0], attributes[1],

attributes[2],

Integer.parseInt(attributes[3]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(vertices);

return builder.toString();

}

public int getVertices() {

return vertices;

}

public void setVertices(int vertices) {

this.vertices = vertices;

}

}

POLYHEDRAFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PolyhedraForm extends SolidForm {

private static final long serialVersionUID = 1L;

private JTextField vertices;

public PolyhedraForm() {

super();

this.add(new JLabel("Vertices:"));

this.vertices = new JTextField();

this.add(this.vertices);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(vertices.getText());

return result;

}

}

PRISM.JAVA

package geofig.shapes;

public class Prism extends Polyhedra {

private String crossSection;

public Prism() {

}

public Prism(String name, String description, String surfaceArea,

int vertices, String crossSection) {

super(name, description, surfaceArea, vertices);

this.crossSection = crossSection;

}

public Prism(String[] attributes) {

this(attributes[0], attributes[1],

attributes[2],

Integer.parseInt(attributes[3]),

attributes[4]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(crossSection);

return builder.toString();

}

public String getCrossSection() {

return crossSection;

}

public void setCrossSection(String crossSection) {

this.crossSection = crossSection;

}

}

PRISMFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PrismForm extends PolyhedraForm {

private static final long serialVersionUID = 1L;

private JTextField crossSection;

public PrismForm() {

super();

this.add(new JLabel("Cross section:"));

this.crossSection = new JTextField();

this.add(this.crossSection);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(crossSection.getText());

return result;

}

}

PYRAMID.JAVA

package geofig.shapes;

public class Pyramid extends Polyhedra {

private String base;

public Pyramid() {

}

public Pyramid(String name, String description, String surfaceArea,

int vertices, String base) {

super(name, description, surfaceArea, vertices);

this.base = base;

}

public Pyramid(String[] attributes) {

this(attributes[0], attributes[1],

attributes[2],

Integer.parseInt(attributes[3]),

attributes[4]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(base);

return builder.toString();

}

public String getBase() {

return base;

}

public void setBase(String base) {

this.base = base;

}

}

PYRAMIDFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class PyramidForm extends PolyhedraForm {

private static final long serialVersionUID = 1L;

private JTextField base;

public PyramidForm() {

super();

this.add(new JLabel("Base:"));

this.base = new JTextField();

this.add(this.base);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(base.getText());

return result;

}

}

RAY.JAVA

package geofig.shapes;

public class Ray extends Line {

private float x;

private float y;

private String direction; // complex

public Ray() {

}

public Ray(String name, String description, float x, float y, String direction) {

super(name, description);

this.x = x;

this.y = y;

this.direction = direction;

}

public Ray(String[] attributes) {

this(attributes[0], attributes[1],

Float.parseFloat(attributes[2]),

Float.parseFloat(attributes[3]),

attributes[4]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(x).append(", ");

builder.append(y).append(" | ");

builder.append(direction);

return builder.toString();

}

public float getX() {

return x;

}

public void setX(float x) {

this.x = x;

}

public float getY() {

return y;

}

public void setY(float y) {

this.y = y;

}

public String getDirection() {

return direction;

}

public void setDirection(String direction) {

this.direction = direction;

}

}

RAYFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class RayForm extends LineForm {

private static final long serialVersionUID = 1L;

private JTextField cX;

private JTextField cY;

private JTextField direction;

public RayForm() {

super();

this.add(new JLabel("X:"));

this.cX = new JTextField();

this.add(this.cX);

this.add(new JLabel("Y:"));

this.cY = new JTextField();

this.add(this.cY);

this.add(new JLabel("Direction:"));

this.direction = new JTextField();

this.add(this.direction);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(cX.getText());

result.add(cY.getText());

result.add(direction.getText());

return result;

}

}

REGULARPOLYGON.JAVA

package geofig.shapes;

public class RegularPolygon extends Polygon {

private float interiorAngle;

public RegularPolygon() {

// TODO Auto-generated constructor stub

}

public RegularPolygon(String name, String description, String area,

int sides, float interiorAngle) {

super(name, description, area, sides);

this.interiorAngle = interiorAngle;

}

public RegularPolygon(String[] attributes) {

this(attributes[0], attributes[1], attributes[2],

Integer.parseInt(attributes[3]),

Float.parseFloat(attributes[4]));

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(interiorAngle);

return builder.toString();

}

public float getInteriorAngle() {

return interiorAngle;

}

public void setInteriorAngle(float interiorAngle) {

this.interiorAngle = interiorAngle;

}

}

REGULARPOLYGONFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class RegularPolygonForm extends PolygonForm {

private static final long serialVersionUID = 1L;

private JTextField interiorAngle;

public RegularPolygonForm() {

super();

this.add(new JLabel("Interor angle:"));

this.interiorAngle = new JTextField();

this.add(this.interiorAngle);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(interiorAngle.getText());

return result;

}

}

SHAPE.JAVA

package geofig.shapes;

public class Shape {

private String name;

private String description;

public Shape() {

}

public Shape(String name, String description) {

this.name = name;

this.description = description;

}

public Shape(String[] attributes) {

this(attributes[0], attributes[1]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("\n").append(name).append(" | ");

builder.append(description);

return builder.toString();

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

SHAPEFORM.JAVA

package geofig.shapes.forms;

import java.awt.GridLayout;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class ShapeForm extends JPanel {

private static final long serialVersionUID = 1L;

private JTextField name;

private JTextArea description;

public ShapeForm() {

this.setLayout(new GridLayout(0, 2));

this.add(new JLabel("Name:"));

this.name = new JTextField();

this.add(this.name);

this.add(new JLabel("Description:"));

this.description = new JTextArea();

this.add(this.description);

}

public ArrayList<String> attributes() {

ArrayList<String> result = new ArrayList<>();

result.add(name.getText());

result.add(description.getText());

return result;

}

}

SHAPES.JAVA

package geofig;

public class Shapes {

static final String SHAPE = "Shape";

static final String POINT = "Point";

static final String PLANE = "Plane";

static final String LINE = "Line";

static final String SOLID = "Solid";

static final String POLYGON = "Polygon";

static final String CURVE = "Curve";

static final String RAY = "Ray";

static final String LINE\_SEGMENT = "Line Segment";

static final String NON\_POLYHEDRA = "Non-Polyhedra";

static final String POLYHEDRA = "Polyhedra";

static final String COMPLEX = "Complex";

static final String IRREGULAR\_POLYGON = "Irregular Polygon";

static final String CONCAVE = "Concave";

static final String REGULAR\_POLYGON = "Regular Polygon";

static final String CIRCLE = "Circle";

static final String ELLIPSE = "Ellipse";

static final String CONE = "Cone";

static final String CYLINDER = "Cylinder";

static final String SPHERE = "Sphere";

static final String TORUS = "Torus";

static final String CUBE = "Cube";

static final String PYRAMID = "Pyramid";

static final String PLATONIC\_SOLID = "Platonic Solid";

static final String PRISM = "Prism";

}

SOLID.JAVA

package geofig.shapes;

public class Solid extends Shape {

private String surfaceArea;

public Solid() {

}

public Solid(String name, String description, String surfaceArea) {

super(name, description);

this.surfaceArea = surfaceArea;

}

public Solid(String[] attributes) {

this(attributes[0], attributes[1], attributes[2]);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(surfaceArea);

return builder.toString();

}

public String getSurfaceArea() {

return surfaceArea;

}

public void setSurfaceArea(String surfaceArea) {

this.surfaceArea = surfaceArea;

}

}

SOLIDFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class SolidForm extends ShapeForm {

private static final long serialVersionUID = 1L;

private JTextField surfaceArea;

public SolidForm() {

super();

this.add(new JLabel("Surface area:"));

this.surfaceArea = new JTextField();

this.add(this.surfaceArea);

}

public ArrayList<String> attributes() {

ArrayList<String> result = super.attributes();

result.add(surfaceArea.getText());

return result;

}

}

SPHERE.JAVA

package geofig.shapes;

public class Sphere extends NonPolyhedra {

public Sphere() {

}

public Sphere(String name, String description, String surfaceArea) {

super(name, description, surfaceArea);

}

public Sphere(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

SPHEREFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class SphereForm extends NonPolyhedraForm {

private static final long serialVersionUID = 1L;

public SphereForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}

TORUS.JAVA

package geofig.shapes;

public class Torus extends NonPolyhedra {

public Torus() {

}

public Torus(String name, String description, String surfaceArea) {

super(name, description, surfaceArea);

}

public Torus(String[] attributes) {

super(attributes);

}

@Override

public String toString() {

return super.toString();

}

}

TORUSFORM.JAVA

package geofig.shapes.forms;

import java.util.ArrayList;

public class TorusForm extends NonPolyhedraForm {

private static final long serialVersionUID = 1L;

public TorusForm() {

super();

}

public ArrayList<String> attributes() {

return super.attributes();

}

}