**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

КУРСОВА РАБОТА  
ПО ПРОГРАМНИ ЕЗИЦИ

Студент:

Факултетен номер:

Група:

Специалност:

Дата: Изготвил:  
 Проверил:

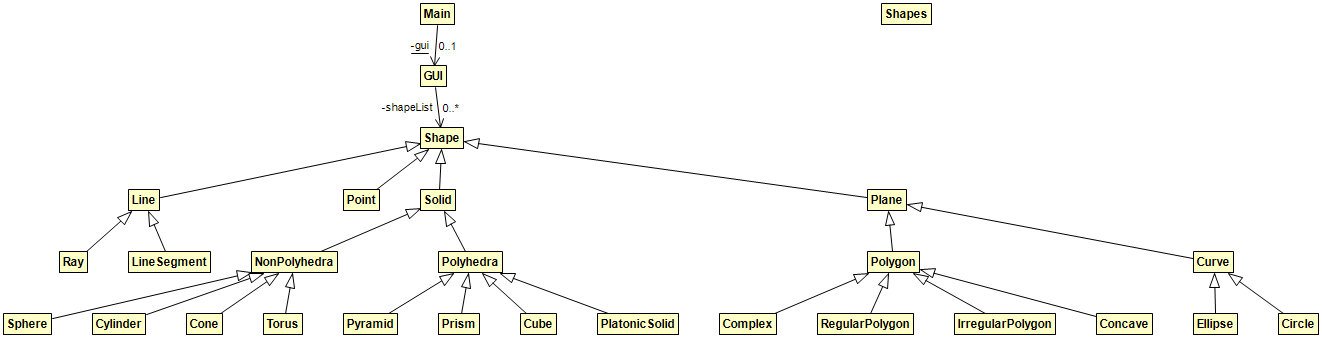
# Задание.

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;

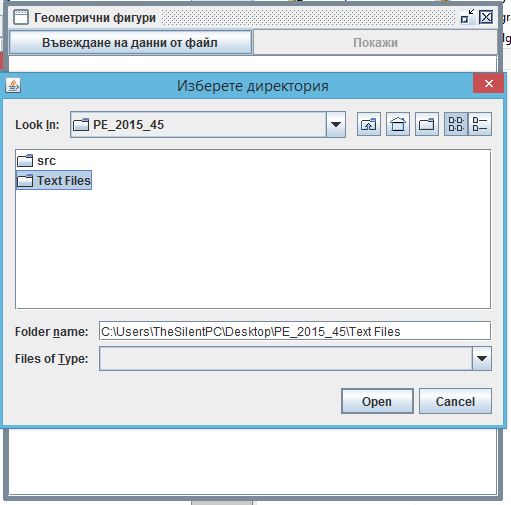
Данните за обектите да се попълват от **текстов файл, като данните за обектите на всеки клас са в отделни файлове (т.е. има толкова файлове, колкото са класовете в йерархията) (форматът им е по избор на студента)** и да се съхраняват в **LinkedList(java.util.LinkedList)**. Изборът на атрибути и методи е на студента, но броят, типовете им и разположението име в йерархията влияят на оценката.

# Клас диаграми.

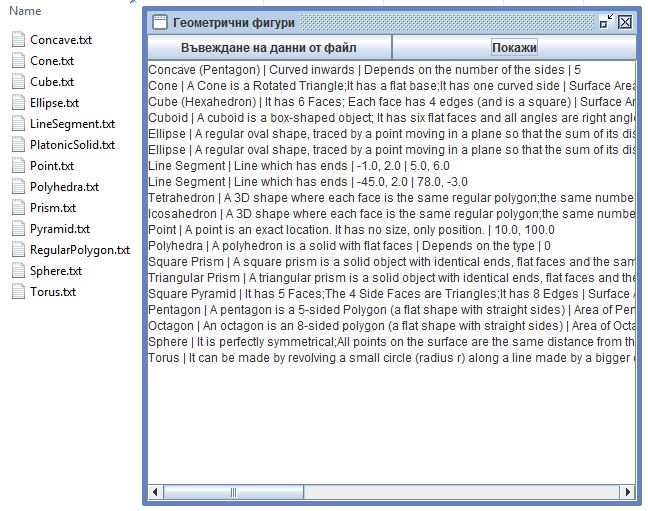


Фигура 1 Клас диаграма.

# Тестови примери



Фигура 2 Избор на файл



Фигура 3 Визуализиране на данните

# 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 geofig.shapes.\*;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FilenameFilter;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.Scanner;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

public class GUI implements ActionListener {

private final String SPLITTER = ":";

private File[] files;

private JTextArea taContent;

private JButton btnFile;

private JButton btnShow;

private JFileChooser fc;

private JFrame frame;

private LinkedList<Shape> shapeList;

private int numberOfAttr;

public GUI(String title) {

shapeList = new LinkedList<>();

JFrame.setDefaultLookAndFeelDecorated(true);

frame = new JFrame(title);

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

frame.getContentPane().setLayout(new BorderLayout());

fc = new JFileChooser();

fc.setCurrentDirectory(new java.io.File("."));

fc.setDialogTitle("Изберете директория");

fc.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

fc.setAcceptAllFileFilterUsed(false);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

JPanel pnlButtons = new JPanel(new GridLayout(0, 2));

btnFile = new JButton("Въвеждане на данни от файл");

btnFile.addActionListener(this);

pnlButtons.add(btnFile);

btnShow = new JButton("Покажи");

btnShow.addActionListener(this);

btnShow.setEnabled(false);

pnlButtons.add(btnShow);

frame.getContentPane().add(pnlButtons, BorderLayout.PAGE\_START);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

JPanel pnlContent = new JPanel(new BorderLayout());

taContent = new JTextArea();

JScrollPane scroll = new JScrollPane (taContent,

JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED, JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

pnlContent.add(scroll, BorderLayout.CENTER);

frame.getContentPane().add(pnlContent, BorderLayout.CENTER);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

frame.pack();

frame.setSize(500, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

private void readFiles() throws Exception {

shapeList.clear();

for (File file : files) {

ArrayList<String[]> objectsAttr = getAttributes(file);

for (String[] item : objectsAttr) {

String fileName = file.getName().substring(0, file.getName().

length() - ".txt".length()).toLowerCase();

switch (fileName) {

case Shapes.CIRCLE:

numberOfAttr = numberOfAttributesOf(Circle.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Circle(item));

break;

case Shapes.COMPLEX:

numberOfAttr = numberOfAttributesOf(Complex.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Complex(item));

break;

case Shapes.CONCAVE:

numberOfAttr = numberOfAttributesOf(Concave.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Concave(item));

break;

case Shapes.CONE:

numberOfAttr = numberOfAttributesOf(Cone.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Cone(item));

break;

case Shapes.CUBE:

numberOfAttr = numberOfAttributesOf(Cube.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Cube(item));

break;

case Shapes.CURVE:

numberOfAttr = numberOfAttributesOf(Curve.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Curve(item));

break;

case Shapes.CYLINDER:

numberOfAttr = numberOfAttributesOf(Cylinder.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Cylinder(item));

break;

case Shapes.ELLIPSE:

numberOfAttr = numberOfAttributesOf(Ellipse.class) - 1;

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Ellipse(item));

break;

case Shapes.IRREGULAR\_POLYGON:

numberOfAttr = numberOfAttributesOf(IrregularPolygon.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new IrregularPolygon(item));

break;

case Shapes.LINE:

numberOfAttr = numberOfAttributesOf(Line.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Line(item));

break;

case Shapes.LINE\_SEGMENT:

numberOfAttr = numberOfAttributesOf(LineSegment.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new LineSegment(item));

break;

case Shapes.NON\_POLYHEDRA:

numberOfAttr = numberOfAttributesOf(NonPolyhedra.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new NonPolyhedra(item));

break;

case Shapes.PLANE:

numberOfAttr = numberOfAttributesOf(Plane.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Plane(item));

break;

case Shapes.PLATONIC\_SOLID:

numberOfAttr = numberOfAttributesOf(PlatonicSolid.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new PlatonicSolid(item));

break;

case Shapes.POINT:

numberOfAttr = numberOfAttributesOf(Point.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Point(item));

break;

case Shapes.POLYGON:

numberOfAttr = numberOfAttributesOf(Polygon.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Polygon(item));

break;

case Shapes.POLYHEDRA:

numberOfAttr = numberOfAttributesOf(Polyhedra.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Polyhedra(item));

break;

case Shapes.PRISM:

numberOfAttr = numberOfAttributesOf(Prism.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Prism(item));

break;

case Shapes.PYRAMID:

numberOfAttr = numberOfAttributesOf(Pyramid.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Pyramid(item));

break;

case Shapes.RAY:

numberOfAttr = numberOfAttributesOf(Ray.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Ray(item));

break;

case Shapes.REGULAR\_POLYGON:

numberOfAttr = numberOfAttributesOf(RegularPolygon.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new RegularPolygon(item));

break;

case Shapes.SHAPE:

numberOfAttr = numberOfAttributesOf(Shape.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Shape(item));

break;

case Shapes.SOLID:

numberOfAttr = numberOfAttributesOf(Solid.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Solid(item));

break;

case Shapes.SPHERE:

numberOfAttr = numberOfAttributesOf(Sphere.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Sphere(item));

break;

case Shapes.TORUS:

numberOfAttr = numberOfAttributesOf(Torus.class);

if (numberOfAttr != item.length) {

throw new Exception();

}

shapeList.add(new Torus(item));

break;

default:

System.out.println("\"" + fileName + "\" has invalid name!");

break;

}

}

}

}

private static int numberOfAttributesOf(Class<?> c\_class) {

int result = 0;

Class<?> current = c\_class;

while(current.getSuperclass() != null) {

result += current.getDeclaredFields().length;

current = current.getSuperclass();

}

return result;

}

@SuppressWarnings("resource")

private ArrayList<String[]> getAttributes(File file) {

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

Scanner fileScanner;

try {

fileScanner = new Scanner(file);

while (fileScanner.hasNextLine()) {

result.add(fileScanner.nextLine().split(SPLITTER));

}

return result;

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return null;

}

private void printObejctsIntoTextArea() {

taContent.setText("");

for (Shape shape : shapeList) {

taContent.append(shape.toString());

}

taContent.setText(taContent.getText().trim());

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnFile) {

int returnVal = fc.showOpenDialog(frame);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

files = fc.getSelectedFile().listFiles(new FilenameFilter() {

public boolean accept(File dir, String name) {

return name.toLowerCase().endsWith(".txt");

}

});

btnShow.setEnabled(true);

try {

readFiles();

} catch (Exception e1) {

e1.printStackTrace();

}

} else if (files == null) {

btnShow.setEnabled(false);

}

} else if (e.getSource() == btnShow) {

printObejctsIntoTextArea();

}

}

}

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 = "linesegment";

static final String NON\_POLYHEDRA = "nonpolyhedra";

static final String POLYHEDRA = "polyhedra";

static final String COMPLEX = "complex";

static final String IRREGULAR\_POLYGON = "irregularpolygon";

static final String CONCAVE = "concave";

static final String REGULAR\_POLYGON = "regularpolygon";

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 = "platonicsolid";

static final String PRISM = "prism";

}

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;

}

}

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();

}

}

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();

}

}

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();

}

}

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();

}

}

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;

}

}

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();

}

}

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;

}

}

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();

}

}

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();

}

}

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;

}

}

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();

}

}

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;

}

}

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();

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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;

}

}

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();

}

}

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();

}

}