Русенски университет „Ангел Кънчев“

**Курсова работа**

по

Програмни езици

Изготвил:

Йордан Йорданов, 26A, 223046

Дата: 19.09.2023 г.

Град: Русе

**Задание**

Да се разработи елементарен векторен графичен редактор на програмния език Java.

* Графичният редактор да позволява избор на цвят за изчертаване и евентуално на цвят за запълване и да поддържа следните графични примитиви: **линия (отсечка), незапълнен четириъгълник, запълнен четириъгълник.**
* Изборът на примитив, който ще се въвежда, да става чрез **меню**.
* Потребителският интерфейс и самите графични примитиви да са разположени в **един прозорец**.
* По време на работа с редактора обектите от графичните примитиви да се съхраняват в масив (по желание на студента може да се използва и колекция, например java.util.ArrayList). В потребителския интерфейс да се предвиди команда за извеждане на съдържанието му в конзола.
* В програмата може да се използват различни класове по усмотрение на студента, но задължително за всеки графичен примитив, зададен по условие, трябва да има отделен клас. Всички тези класове трябва да реализират следния интерфейс:

public interface GPrimitive{

public abstract void drawPrimitive(Graphics g);

// методът изчертава графичен примитив

}

* За базов тип на масива или колекцията трябва да се използва интерфейса GPrimitive, а като елементи да се съхраняват обекти от класовете за отделните фигури (графични примитиви).
* Как се задават примитивите (например четириъгълник може да бъде зададен по две точки или по точка, ширина и височина) и как се въвеждат стойности (например чрез работа с мишката, както в комерсиалните редактори, или чрез въвеждане на числа в някакъв потребителски интерфейс) е по избор на студента.

**Изходен код**

**Github хранилище** => https://github.com/thr33bricks/MiniPaintA qr code with a lion head

Description automatically generated

**GPrimitive.java**

package com**.**yordanyordanov**.**kursova**;**

**import** java**.**awt**.**Graphics**;**

public interface GPrimitive **{**

//draws figure

public abstract void drawPrimitive**(**Graphics g**);**

**}**

**TwoPointFigure.java**

package com**.**yordanyordanov**.**kursova**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Graphics**;**

**import** java**.**text**.**MessageFormat**;**

public class TwoPointFigure **implements** GPrimitive **{**

int x1 **=** 0**;**

int y1 **=** 0**;**

int x2 **=** 0**;**

int y2 **=** 0**;**

Color col **=** Color**.**black**;**

TwoPointFigure**(){}**

TwoPointFigure**(**int x1**,** int y1**,** int x2**,** int y2**,** Color c**){**

**this.**x1 **=** x1**;**

**this.**y1 **=** y1**;**

**this.**x2 **=** x2**;**

**this.**y2 **=** y2**;**

col **=** c**;**

**}**

public boolean isInsideDrawArea**()** **{**

**return** **false;**

**}**

@Override

public void drawPrimitive**(**Graphics g**)** **{}**

public TwoPointFigure Copy**()** **{**

TwoPointFigure f **=** **new** TwoPointFigure**(**x1**,** y1**,** x2**,** y2**,** col**);**

**return** f**;**

**}**

@Override

public String toString**()** **{**

**return** MessageFormat**.**format**(**"TwoPointFigure(x1={0}, y1={1}, x2={2}, y2={3}, Colour=[{4},{5},{6}])"**,**

x1**,** y1**,** x2**,** y2**,** col**.**getRed**(),** col**.**getGreen**(),** col**.**getBlue**());**

**}**

**}**

**Line.java (частично)**

package com**.**yordanyordanov**.**kursova**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Graphics**;**

**import** java**.**text**.**MessageFormat**;**

public class Line **extends** TwoPointFigure **implements** GPrimitive**{**

final double error **=** 0.3**;**

Line**(){}**

Line**(**int x1**,** int y1**,** int x2**,** int y2**,** Color c**){**

**super(**x1**,** y1**,** x2**,** y2**,** c**);**

**}**

@Override

public boolean isInsideDrawArea**(){** **}**

@Override

public void drawPrimitive**(**Graphics g**)** **{**

Color prevCol **=** g**.**getColor**();**

g**.**setColor**(**col**);**

g**.**drawLine**(**x1**,** y1**,** x2**,** y2**);**

g**.**setColor**(**prevCol**);**

**}**

@Override

public Line Copy**()** **{**

Line l **=** **new** Line**(**x1**,** y1**,** x2**,** y2**,** col**);**

**return** l**;**

**}**

private double triangleArea**(**int x1**,** int y1**,** int x2**,** int y2**,** int x3**,** int y3**){**

**return** Math**.**abs**((**x1**\*((**double**)**y2**-**y3**)** **+** x2**\*((**double**)**y3**-**y1**)+** x3**\*((**double**)**y1**-**y2**))/**2.0**);**

**}**

// Triangle defined by two points and third is max x and max y of the provided

boolean isInsideTriangle**(**int x1**,** int y1**,** int x2**,** int y2**,** int x**,** int y**){** **}**

@Override

public String toString**()** **{}**

**}**

**Rectangle.java**

package com**.**yordanyordanov**.**kursova**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Graphics**;**

**import** java**.**text**.**MessageFormat**;**

public class Rectangle **extends** TwoPointFigure **implements** GPrimitive**{**

Rectangle**(){}**

Rectangle**(**int x1**,** int y1**,** int x2**,** int y2**,** Color c**){**

**super(**x1**,** y1**,** x2**,** y2**,** c**);**

**}**

@Override

public boolean isInsideDrawArea**(){**

//first point is visible

**return** **(**x1 **<** MiniPaint**.**currWidth **&&** y1 **<** MiniPaint**.**currHeight **-** MiniPaint**.**footerHeight**)** **||**

**(**x2 **<** MiniPaint**.**currWidth **&&** y2 **<** MiniPaint**.**currHeight **-** MiniPaint**.**footerHeight**);**

**}**

@Override

public void drawPrimitive**(**Graphics g**)** **{**

Color prevCol **=** g**.**getColor**();**

g**.**setColor**(**col**);**

g**.**drawRect**(**Math**.**min**(**x1**,** x2**),** Math**.**min**(**y1**,** y2**),** Math**.**abs**(**x1**-**x2**),** Math**.**abs**(**y1**-**y2**));**

g**.**setColor**(**prevCol**);**

**}**

@Override

public Rectangle Copy**()** **{**

Rectangle r **=** **new** Rectangle**(**x1**,** y1**,** x2**,** y2**,** col**);**

**return** r**;**

**}**

@Override

public String toString**()** **{**

**return** MessageFormat**.**format**(**

"Rectangle(x1={0}, y1={1}, x2={2}, y2={3}, Colour=[{4},{5},{6}])"**,**

x1**,** y1**,** x2**,** y2**,** col**.**getRed**(),** col**.**getGreen**(),** col**.**getBlue**());**

**}**

**}**

**FillRectangle.java**

Кодът е идентичен на Rectangle.java, единствено e смененa ключовата дума Rectangle с FillRectangle и функцията drawPrimitive използва метода за изчертаване на запълнен правоъгълник.

**MiniPaint.java**

Това е файла с основната логика на програмата. Там се обработват събитията, случващи се в графичния интерфейс.