Донецкий Национальный Технический Университет

Лабораторная работа № 1 - 4

Моделирование

Выполнил:

ст. группы ИПЗ -13

Лысенко А. С.

Проверила:

Скрипник. Т. В.

Красноармейск 2016

Base.java

**package** Source;  
**public abstract class** Base  
{  
 **public abstract** String func(String s);  
}

Factory.java

**package** Source;  
**public class** Factory **extends** Base  
{  
 **public** Factory(String s1)  
 {  
 String s = s1;  
 **this**.func(s);  
 }  
  
 @Override  
 **public** String func(String s)  
 {  
 **return** s;  
 }  
}

DrawLine.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** DrawLine  
{  
 **public abstract void** drawLine(Graphics g);  
}

LineDraw.java

**package** Source;  
**import** java.awt.\*;  
**public class** LineDraw **extends** DrawLine  
{  
 **public** LineDraw(Graphics g)  
 {  
 **this**.drawLine(g);  
 }  
 @Override  
 **public void** drawLine(Graphics g)  
 {  
 g.drawLine(590,50,650,50);  
 g.drawLine(650,50,650,40);  
 g.drawLine(650,40,590,40);  
 g.drawLine(590,40,590,50);  
 }  
}

DrawText.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** DrawText  
{  
 **public abstract void** fDraw(Graphics g);  
}

ExDraw.java

**package** Source;  
**import** java.awt.\*;  
**public class** ExDraw **extends** DrawText  
{  
 **public** ExDraw(Graphics g)  
 {  
 **this**.fDraw(g);  
 }  
 @Override  
 **public void** fDraw(Graphics g)  
 {  
 g.drawString(**"График"**, 600,50);  
 }  
}

LineToDraw.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** LineToDraw  
{  
 **public abstract void** lineToDraw(Graphics g);  
}

DrawToLine.java

**package** Source;  
**import** java.awt.\*;  
**public class** DrawToLine **extends** LineToDraw  
{  
 **public** DrawToLine(Graphics g)  
 {  
 **this**.lineToDraw(g);  
 }  
 @Override  
 **public void** lineToDraw(Graphics g)  
 {  
 **new** LineDraw(g).drawLine(g);  
 }  
}

Frame.java

**package** Source;  
**import** javax.swing.\*;  
**public class** Frame **extends** JFrame  
{  
 **public** Frame()  
 {  
 setContentPane(**new** GUI());  
 setTitle(**"График"**);  
 setSize(700,500);  
 setLocationRelativeTo(**null**);  
 setResizable(**false**);  
 setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 setVisible(**true**);  
 }  
}

GUI.java

**package** Source;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**public class** GUI **extends** JPanel  
{  
 **private int ny** , **nx** , **oyn** , **oyk** , **oyx** , **oxn** , **oxk** , **oxy** , **ly** , **lx** , **sw** , **xln** , **l2**;  
 **private float xng** , **kx** , **ky** , **hx** , **yg** , **xk**;  
 **int i**;  
 **int**[] **mas**;  
 **int x**;  
 **int x1**;  
 String **f**;  
 **char**[] **ch**;  
 **boolean fl** = **false**;  
 String **s1**,**s2**,**s3**;  
 **float um**;  
 **float chM**;  
 **float chP**;  
 **float chX**;  
 **float del**;  
 **float plus**;  
 **float minus**;  
 **double pow**;  
 **double xInt** = 0;  
 **int d** = 0;  
 String **s**;  
 JLabel **l** = **new** JLabel(**"f(x) = "**);  
 JLabel **l1** = **new** JLabel(**"x: "**);  
 JButton **b** = **new** JButton(**"Подтвердить"**);  
 JButton **b1** = **new** JButton(**"Очистить"**);  
 JTextField **t** = **new** JTextField(15);  
 JTextField **t1** = **new** JTextField(5);  
  
 **public** GUI()  
 {  
 add(**l**);  
 add(**t**);  
 add(**l1**);  
 add(**t1**);  
 add(**b**);  
 add(**b1**);  
 **ny** = 10;*// цена деления по шкалам* **ky** = (**float**) 0.5; *// коэф шкалы по у* **kx** = (**float**) 0.5; *// коэф шкалы по x* **oyn** = 50; *// начальный отступ по y* **oxn** = 50; *//начальный отступ по х* **ly** = 400; *// длина оси у* **lx** = 600; *// длина оси х  
 // по умолчанию в начале на экран выводится график y=x* **hx** = (**float**) 0.011;*//шаг табуляции* **b**.addActionListener(**new** AbstractAction()  
 {  
 Graphics **g**;  
 **public void** actionPerformed(ActionEvent e)  
 {  
 **s** = **t**.getText();  
 **this**.enterPoint();  
 **this**.enterFunc();  
 **g** = getGraphics();  
 funcLine(**g**);  
 **new** Wrapper(**g**);  
 **new** DrawToLine(**g**);  
 **new** StrategyGo(**g**);  
 **new** StrategyGo\_1(**g**);  
 }  
 **public void** enterPoint()  
 {  
 **x** = **new** Integer(**t1**.getText());  
 **x1** = (**int**) (**x** \* **xInt**);  
  
 }  
 **public void** enterFunc()  
 {  
 Factory f = **new** Factory(**t**.getText());  
 **ch** = f.func(**t**.getText()).trim().toCharArray();  
 **for** (**i** = 0; **i** < **ch**.**length** - 1; **i**++) {  
 **if** (**ch**[**i**] == **' '**)  
 **d**++;  
 **else if** (**ch**[**i**] == **'\*'** && **ch**[**i** + 3] == **'x'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **s3** = Character.*toString*(**ch**[**i** + 3]);  
 funcInt(**s3**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **um** = **chM** \* **chP** \* **x**;  
 **xInt** = **um**;  
 } **else if** (**ch**[**i**] == **'/'** && **ch**[**i** + 3] == **'x'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **s3** = Character.*toString*(**ch**[**i** + 3]);  
 funcInt(**s3**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **x** = (**int**)**chX**;  
 **del** = **chM** / **chP** \* **x**;  
 **xInt** = **del**;  
 } **else if** (**ch**[**i**] == **'+'** && **ch**[**i** + 3] == **'x'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **s3** = Character.*toString*(**ch**[**i** + 3]);  
 funcInt(**s3**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **x** = (**int**)**chX**;  
 **plus** = **chM** + **chP** \* **x**;  
 **xInt** = **plus**;  
 } **else if** (**ch**[**i**] == **'-'** && **ch**[**i** + 3] == **'x'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **s3** = Character.*toString*(**ch**[**i** + 3]);  
 funcInt(**s3**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **x** = (**int**)**chX**;  
 **minus** = **chM** - **chP** \* **x**;  
 **xInt** = **minus**;  
 }  
 **if** (**ch**[**i**] == **'^'** && **ch**[**i** + 3] == **'x'**)  
 {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **s3** = Character.*toString*(**ch**[**i** + 3]);  
 funcInt(**s3**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **x** = (**int**)**chX**;  
 **pow** = Math.*pow*(**chM**, **chP** \* **x**);  
 **xInt** = **pow**;  
 }  
 }  
 }  
 **int** funcInt(String s)  
 {  
 **try** {  
 **fl** = **true**;  
 **return new** Integer(s);  
 } **catch** (NumberFormatException e)  
 {  
 **fl** = **false**;  
 s = **t**.getText();  
 **return** 0;  
 }  
 }  
 });  
 **b1**.addActionListener(**new** ActionListener()  
 {  
 @Override  
 **public void** actionPerformed(ActionEvent e)  
 {  
 repaint();  
 }  
 });  
 }  
 **public void** paint(Graphics g)  
 {  
 **super**.paint(g);  
 *//funcLine(g);  
 //Разбиваем каждую ось на две части для удобства переноса центра координат  
 // Ось Y* g.drawLine((**int**) (**lx** \* **kx** + **oxn**), **oyn**,  
 (**int**) (**lx** \* **kx** + **oxn**), **ly** + **oyn**);  
 *// Стрелки* g.drawLine((**int**) (**lx** \* **kx** + **oxn**), **oyn**,  
 (**int**) (**lx** \* **kx** + **oxn**) - 3, **oyn** + 10);  
 g.drawLine((**int**) (**lx** \* **kx** + **oxn**), **oyn**,  
 (**int**) (**lx** \* **kx** + **oxn**) + 3, **oyn** + 10);  
 *// Надпись* g.drawString(**"Y"**, (**int**) (**lx** \* **kx** + **oxn**) - 10, **oyn** + 10);  
 g.drawString(**"0"**, (**int**) (**lx** \* **kx** + **oxn**) - 9, (**int**) (**ly** \* **ky** + **oyn**) + 11);  
 *//Деления* **int** l1 = (**int**) (**ly** \* **ky**);  
 **l2** = **ly** - l1;  
 **int** k1 = (**int**) l1 / **ny**;  
 **int** k2 = (**int**) **l2** / **ny**;  
 **for** (**int** i = 1; i < k1 + 1; i++) {  
 g.drawLine((**int**) (**lx** \* **kx** - 2 + **oxn**), l1 - **ny** + **oyn**,  
 (**int**) (**lx** \* **kx** + 2 + **oxn**), l1 - **ny** + **oyn**);  
 l1 = l1 - **ny**;  
 }  
 l1 = **ly** - **l2**;  
 **for** (**int** i = 1; i < k2 + 1; i++) {  
 g.drawLine((**int**) (**lx** \* **kx** - 2 + **oxn**), l1 + **ny** + **oyn**,  
 (**int**) (**lx** \* **kx** + 2 + **oxn**), l1 + **ny** + **oyn**);  
 l1 = l1 + **ny**;  
 }  
 *// Ось Х* g.drawLine(**oxn**, (**int**) (**ly** \* **ky** + **oyn**), **lx** + **oxn**, (**int**) (**ly** \* **ky** + **oyn**));  
 g.drawLine(**lx** + **oxn**, (**int**) (**ly** \* **ky** + **oyn**), **lx** + **oxn** - 10,  
 (**int**) (**ly** \* **ky** + **oyn**) - 3);  
 g.drawLine(**lx** + **oxn**, (**int**) (**ly** \* **ky** + **oyn**), **lx** + **oxn** - 10,  
 (**int**) (**ly** \* **ky** + **oyn**) + 3);  
 *// Надпись* g.drawString(**"Х"**, **lx** + **oyn** - 10, (**int**) (**ly** \* **ky** + **oyn**) - 5);  
 *// Деления* l1 = (**int**) (**lx** \* **kx**);  
 **l2** = **lx** - l1;  
 k1 = l1 / **ny**;  
 k2 = **l2** / **ny**;  
 **for** (**int** i = 1; i < k1 + 1; i++) {  
 g.drawLine(l1 - **ny** + **oxn**, (**int**) (**ly** \* **ky** - 2 + **oyn**),  
 l1 - **ny** + **oxn**, (**int**) (**ly** \* **ky** + 2 + **oyn**));  
 l1 = l1 - **ny**;  
 }  
 l1 = **lx** - **l2**;  
 **double** xl = l1 / **ny**;  
 **double** xl1 = **l2** / **ny**;  
 **for** (**int** i = 1; i < k2 + 1; i++)  
 {  
 g.drawLine(l1 + **ny** + **oxn**, (**int**) (**ly** \* **ky** - 2 + **oyn**),  
 l1 + **ny** + **oxn**, (**int**) (**ly** \* **ky** + 2 + **oyn**));  
 l1 = l1 + **ny**;  
 }  
 }  
 **void** funcLine(Graphics g)  
 {  
 **xk** = 0 ;  
 **yg** = 0;  
 **while**( ( **xk** + **hx** ) \* **ny** < **lx** && ( **xk** + **hx** ) \* **ny** < **ly** \* **ky** )  
 {  
 **yg** = **xk** ;  
 g.drawLine( (**int**) (**lx** \* **kx** + **oxn**),  
 (**int**) (**ly** \* **ky** + **oyn**), **x** ,**x1**);  
 **xk** = **xk** + **hx** ;  
 }  
 **xk** = 0 ;  
 **yg** = 0;  
 **while**( ( **xk** + **hx** ) \* **ny** < **lx** && ( **xk** + **hx** ) \* **ny** < **lx** \* **kx** )  
 {  
 **yg** = **xk** ;  
 g.drawLine( (**int**) (**lx** \* **kx** + **oyn**),  
 (**int**) (**ly** \* **ky** + **oxn**), **x** ,**x1**);  
 **xk** = **xk** + **hx** ;  
 }  
 }  
}

Logic.java

**package** Source;  
**import** java.util.Scanner;  
**public class** Logic  
{  
 Scanner **scn** = **new** Scanner(System.***in***);  
 **int i**;  
 **int**[] **mas**;  
 **int x** = 0;  
 String **f**;  
 **char**[] **ch**;  
 **boolean fl** = **false**;  
 String **s1**,**s2**;  
 **float um**;  
 **float chM**;  
 **float chP**;  
 **float del**;  
 **float plus**;  
 **float minus**;  
 **double pow**;  
 **double xInt** = 0;  
 **int d** = 0;  
 String **s**;  
 **public** Logic(String str)  
 {  
 **s** = str;  
 **this**.enterFunc();  
 **this**.enterPoint();  
 }  
 **public void** enterFunc()  
 {  
 System.***out***.print(**"Введите функцию f(x): "**);  
 **f** = **scn**.nextLine();  
 **ch** = **f**.trim().toCharArray();  
 **for** (**i** = 0; **i** < **ch**.**length** - 1; **i**++) {  
 **if** (**ch**[**i**] == **' '**)  
 **d**++;  
 **else if** (**ch**[**i**] == **'\*'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **um** = **chM** \* **chP**;  
 **xInt** = **um**;  
 System.***out***.println(**chM** + **" \* "** + **chP** + **" = "** + **um**);  
 } **else if** (**ch**[**i**] == **'/'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **del** = **chM** / **chP**;  
 **xInt** = **del**;  
 System.***out***.println(**chM** + **" / "** + **chP** + **" = "** + **del**);  
 } **else if** (**ch**[**i**] == **'+'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **plus** = **chM** + **chP**;  
 **xInt** = **plus**;  
 System.***out***.println(**chM** + **" + "** + **chP** + **" = "** + **plus**);  
 } **else if** (**ch**[**i**] == **'-'**) {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **minus** = **chM** - **chP**;  
 **xInt** = **minus**;  
 System.***out***.println(**chM** + **" - "** + **chP** + **" = "** + **minus**);  
 }  
 **if** (**ch**[**i**] == **'^'**)  
 {  
 **s1** = Character.*toString*(**ch**[**i** - 2]);  
 funcInt(**s1**);  
 **s2** = Character.*toString*(**ch**[**i** + 2]);  
 funcInt(**s2**);  
 **chM** = **new** Integer(**s1**);  
 **chP** = **new** Integer(**s2**);  
 **pow** = Math.*pow*(**chM**, **chP**);  
 **xInt** = **pow**;  
 System.***out***.println(**chM** + **" ^ "** + **chP** + **" = "** + **pow**);  
 }  
 }  
 }  
 **public void** enterPoint()  
 {  
 System.***out***.print(**"Введите кол-во точек x: "**);  
 **x** = **scn**.nextInt();  
 **mas** = **new int**[**x**];  
 **for**(**i** = 0; **i** < **x**; **i**++)  
 {  
 System.***out***.print(**"Введите точку["** + (**i** + 1) + **"]: "**);  
 **mas**[**i**] = **scn**.nextInt();  
 }  
 **for**( **i** = 0; **i** < **x**; **i**++)  
 System.***out***.println(**"Ваша точка["**+ (**i** + 1) + **"]: "** + **mas**[**i**] \* **xInt**);  
 }  
 **int** funcInt(String s)  
 {  
 **try** {  
 **fl** = **true**;  
 **return new** Integer(s);  
 } **catch** (NumberFormatException e)  
 {  
 **fl** = **false**;  
 s = **scn**.next();  
 **return** 0;  
 }  
 }  
}

Source.java

**package** Source;  
**public class** Source  
{  
 **private static** Source *instance* = **null**;  
 **public static void** main(String[] args)  
 {  
 *getInstance*();  
 }  
 **private** Source()  
 {  
 **new** Frame();  
 }  
 **private static** Source getInstance()  
 {  
 **if**(*instance* == **null**)  
 *instance* = **new** Source();  
 **return** *instance*;  
 }  
}

Square.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** Square  
{  
 **public abstract void** wrapper(Graphics g);  
}

Strategy.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** Strategy  
{  
 **public abstract void** showThis(Graphics g);  
}

Strategy\_1.java

**package** Source;  
**import** java.awt.\*;  
**public abstract class** Strategy\_1  
{  
 **public abstract void** showThis\_1(Graphics g);  
}

StratagyGo.java

**package** Source;  
**import** java.awt.\*;  
**public class** StrategyGo **extends** Strategy  
{  
 **public** StrategyGo(Graphics g)  
 {  
 **this**.showThis(g);  
 }  
 @Override  
 **public void** showThis(Graphics g)  
 {  
 g.drawLine(620,80,680,80);  
 g.drawLine(620,80,650,90);  
 g.drawLine(650,90,680,80);  
 }  
}

StratagyGo\_1.java

**package** Source;  
**import** java.awt.\*;  
**public class** StrategyGo\_1 **extends** Strategy\_1  
{  
 **public** StrategyGo\_1(Graphics g)  
 {  
 **this**.showThis\_1(g);  
 }  
  
 @Override  
 **public void** showThis\_1(Graphics g)  
 {  
 g.drawLine(630,100,660,90);  
 g.drawLine(660,90,690,100);  
 g.drawLine(630,100,690,100);  
 }  
}

Wrapper.java

**package** Source;  
**import** java.awt.\*;  
**public class** Wrapper **extends** Square  
{  
 **public** Wrapper(Graphics g)  
 {  
 **this**.wrapper(g);  
 }  
 @Override  
 **public void** wrapper(Graphics g)  
 {  
 **new** ExDraw(g).fDraw(g);  
 }  
}