Лабораторная работа №6. Потоки. Графика

Теоретическая часть

В отличие от консольных приложений оконные приложения представлены на экране прикладным окном, в области клиента которого рисуется графика и размещаются управляющие интерфейсные элементы. Специальный пакет java.awt включает множество классов для создания и применения окон, интерфейсных элементов и графических объектов. Корневым классом в AWT-иерархии является класс Component, включающий множество функций и свойств, используемых в объектах его подклассов Container, Window, Frame и Panel. Среди AWT-классов есть широко применяемые класс Button кнопки, класс Choice раскрывающегося списка, класс Color цвета, класс Font шрифта, класс Frame стандартного окна, класс Image графического изображения, класс Menu меню, класс Point точки, класс Rectangle прямоугольника, класс TextField редактора текста. Среди множества AWT-классов имеются классы, позволяющие разнообразно реализовать как оконные события, так и события интерфейсных оконных элементов.

Графика

Богатый набор функций рисования класса Graphics можно получить, воспользовавшись ссылкой на объект этого класса в параметре функции paint(). Объект типа Graphics содержит состояние графического контекста – текущий цвет, шрифт, текущее положение курсора и др.

В нижеследующей программе в прикладном окне рисуются линия, прямоугольник, эллипс и текст.

```
/* Окно с графикой*/
import java.awt.*;
class Graf extends Frame {
 public void paint (Graphics g) {
  g.drawLine (5, 35, 100, 35); // Нарисовать линию
  g.setColor (Color.blue); // Установить синий цвет
  g.drawRect (5, 40, 40, 60); // Нарисовать прямоугольник
  g.setColor (Color.red); // Установить красный цвет
  g.fillOval (70, 40, 40, 60); // Нарисовать заполненный эллипс
  Font f= new Font ("Verdana", Font.BOLD, 30);
  g.setFont (f);
  // Установить шрифт
  g.drawString ("KAI", 5, 130); // Нарисовать текст
 }
 public static void main (String[] args) {
  Graf gr= new Graf ( ); // Создать объект окна
  gr.setSize (100, 150); // Установить размеры окна
  gr.show ( ); // Показать окно
 }
}
/* Result: В прикладном окне рисуются линия, прямоугольник, эллипс и текст. При попытке закрыть прикладное окно не закрывается.
*/
```

Класс Graf окна программы наследует класс Frame, включающий множество свойств и функций. Функция paint() перерисовки окна класса Frame рисует линию, прямоугольник, эллипс и текст, вызывая соответствующие функции рисования класса Graphics.

Окно и события

Предыдущее простое приложение обладает существенным недостатком – его прикладное окно не закрывается при нажатии на кнопку закрытия окна, размещённую в верхнем правом углу. Нажатие на кнопку не обрабатывается. В следующей программе переопределён обработчик handleEvent () событий. В нём обрабатывается сообщение WINDOW_DESTROY, генерируемое операционной системой при закрытии окна. Обработка других событий адресуется обработчику handleEvent(e) суперкласса.

```
/* Обработка события закрытия окна*/
import java.awt.*;
class Graf2 extends Frame {
 public void paint (Graphics g) {
  g.setColor(Color.red);
  g.fillOval(70, 40, 40, 60);}
  public boolean handleEvent (Event e) {
   if (e.id == Event.WINDOW_DESTROY) System.exit(0);
   return (super.handleEvent(e));
  }
  public static void main (String[] args) {
   Graf2 gr= new Graf2();
   gr.setSize(100, 150); gr.show();
 }
}
/*
Result:
В прикладном окне рисуется эллипс.
При попытке закрыть прикладное окно оно закрывается.
*/
```

Следующая программа для закрытия окна применяет объект класса OurWindowAdapter, наследующий класс адаптера окна WindowAdapter, включающий переопределённую предопределённую функцию windowClosing(), в которой вызывается функция exit() класса. System, завершающая выполнение приложения. В соответствии с правилами обработки событий языка Java функция addWindowListener() подписывает (делегирует) объект класса OurWindowAdapter к объекту источнику класса Window. При возникновении события закрытия окна выполнится делегированная предопределённая функция windowClosing().

```
/* Обработка события закрытия окна*/
import java.awt.*;
import java.awt.event.*;
class Graf3 extends Frame {
 public Graf3 () {
  // Подписаться на обработчик закрытия окна
  this.addWindowListener (new OurWindowAdapter());
 }
 public void paint (Graphics g) {
  g.setColor(Color.red);
  g.fillOval(70, 40, 40, 60);
 }
 public static void main (String[] args) {
  Graf3 gr= new Graf3();
  gr.setSize(100, 150);
  gr.show();}
 }
 class OurWindowAdapter extends WindowAdapter {
  public void windowClosing (WindowEvent wE) {System.exit (0);}
}
/*Result:
В прикладном окне рисуется эллипс.
При попытке закрыть прикладное окно оно закрывается.
*/
//
//-----------------------------------------------------------------------------------------------------------
//
/* Обработка события закрытия окна с использованием анонимного класса*/
import java.awt.*;
import java.awt.event.*;
class Graf4 extends Frame {
 public Graf4() {
 // Подписаться на объект внутреннего анонимного класса
 this.addWindowListener (new WindowAdapter() {
   public void windowClosing (WindowEvent wE){System.exit (0);}
  });
 }
 public void paint (Graphics g) {
  g.setColor(Color.red);
  g.fillOval(70, 40, 40, 60);
 }
 public static void main (String[] args) {
  Graf4 gr = new Graf4();
  gr.setSize(100, 150);
  gr.show();
 }
}
/*Result:
В прикладном окне рисуется эллипс.
При попытке закрыть прикладное окно оно закрывается.
*/
//
```

```
//-------------------------------------------------------------------------------------------------------
//
```
/* Обработка событий мыши и закрытия окна*/
```
import java.awt.*;
import java.awt.event.*;
class Graf5 extends Frame {
 String str = null;
 public Graf5() {
  str = "";
  // Подписаться на объект внутреннего анонимного класса (для мыши)
  this.addMouseListener(new MouseAdapter() {
   public void mousePressed (MouseEvent mE) {
    str = "x= " + mE.getX() + " y= " + mE.getY();
    repaint();
   }
  });
  // Подписаться на объект внутреннего анонимного класса (для окна)
  this.addWindowListener (new WindowAdapter() {
   public void windowClosing (WindowEvent wE){System.exit (0);}
  });
 }
 public void paint (Graphics g) {
  g.setColor(Color.red);
  g.fillOval(70, 40, 40, 60);
  g.drawString(str, 5, 120);
 }
 public static void main (String[] args) {
  Graf5 gr= new Graf5();
  gr.setSize(100, 150);
  gr.show();
 }
}
/*Result:
В прикладном окне рисуется эллипс.
При нажатии на клавишу мыши выдаются координаты носика.
При попытке закрыть прикладное окно оно закрывается.
*/
```

Окно и управляющие интерфейсные элементы

/* Окно, кнопка и редактор текста*/
```
import java.awt.*;
import java.awt.event.*;
class CTextBox_Button extends Frame {// Класс прикладного окна
 TextField ourTextBox; // Ссылка на редактор текста
 Button ourButton; // Ссылка на кнопку
 Point point; // Начальные координаты строки
 Color[] colors; // Ссылка на массив цветов флага
 int n; // Индекс массива цветов colors
 public CTextBox_Button() { // Конструктор
  setSize (new Dimension (400, 200));
  this.setBackground (Color.lightGray);
```

```
 setLayout (new FlowLayout ( ));
 ourTextBox= new TextField ( ); // Создать редактор
 ourTextBox.setSize (150, 20); // Установить размер
 ourTextBox.setText ("Russia"); // Установить текст в редакторе
 add (ourTextBox); // Добавить редактор в форму
 ourButton= new Button ("OK"); // Создать кнопку
 add (ourButton); // Добавить кнопку к форме
 // Подписать обработчик на событие кнопки
 ourButton.addActionListener (new ActionListener() {
  public void actionPerformed (ActionEvent aE) {
   n++; if (n >= 3) n= 0;
  }
 });
 // Подписать обработчик на событие мыши
 this.addMouseListener (new MouseAdapter() {
  public void mousePressed (MouseEvent mE) {
   System.out.println ("Mouse: x = " + mE.getX() + " y= " + mE.getY());
   point.x = mE.getX(); point.y = mE.getY();
   repaint();
  }
 });
 // Подписать обработчик для закрытия окна
 this.addWindowListener (new WindowAdapter() {
  public void windowClosing (WindowEvent wE) {
            System.exit (0);
  }
 });
 n= 0;
 point= new Point (100, 100);
 colors = new Color[3]; // Создать массив цветов
 colors[0]= Color.white; // Серый цвет
 colors[1]= Color.blue; // Синий цвет
 colors[2]= Color.red; // Красный цвет
}
// Перерисовать область клиента окна
public void paint (Graphics g) {// Нарисовать строку от носика курсора мыши
 g.setColor (colors[n]);
 g.drawString (ourTextBox.getText(), point.x, point.y);
}
public static void main (String[] args) {
 CTextBox_Button cT= new CTextBox_Button();
 cT.show();
}
}
```

Окно и дочерние окна

```
/* Окно с дочерним окном*/
import java.awt.*;
import java.awt.event.*;
class CTwoFrames extends Frame {
 Frame childFrame ; // Дочерняя форма
```

```
TextField ourTextBox; // Редактор
Button ourButton; // Кнопка
Point point; // Начальные координаты строки
Color [] colors; // Массив цветов флага
int n; // Индекс массива цветов colors
public CTwoFrames() {
 setTitle("Russian flag");
 setSize (400, 200);
 // Установить заголовок прикладного окна63
 this.setBackground(Color.lightGray);
 ourTextBox= new TextField();
 // Создать редактор
 ourTextBox.setSize(new Dimension(150, 20)); // Установить позицию
 ourTextBox.setText ("Russia"); // Установить текст в редакторе
 ourButton = new Button ("OK"); // Создать кнопку
 // Подписать обработчик на событие кнопки
 ourButton.addActionListener(new ActionListener() {
  public void actionPerformed (ActionEvent aE) {
   n++;
   if (n >= 3) n= 0;
  }
 });
 // Подписать обработчик на событие MouseDown мыши
 this.addMouseListener(new MouseAdapter() {
  public void mousePressed (MouseEvent mE) {
   point.x = mE.getX();
   point.y = mE.getY();
   repaint ( );
  }
 });
 // Подписать обработчик для закрытия окна
 this.addWindowListener (new WindowAdapter() {
  public void windowClosing (WindowEvent wE) {
            System.exit(0);
  }
 });
 n= 0;
 point= new Point (100, 100);
 colors= new Color [3]; // Создать массив цветов
 colors [0]= Color.white; // Белый цвет
 colors [1]= Color.blue; // Синий цвет
 colors [2]= Color.red; // Красный цвет
 childFrame= new Frame(); // Создать дочернюю форму и
 childFrame.setSize (200, 100); // с размером
 childFrame.setLocation (410, 0);
 childFrame.setLayout (new FlowLayout ( ));
 childFrame.add (ourTextBox); // Добавить редактор
 childFrame.add (ourButton); // Добавить кнопку
 childFrame.show();
}
// Перерисовать область клиента окна
public void paint (Graphics g) {// Нарисовать строку от носика курсора мыши
```

```
  g.setColor(colors[n]);
  g.drawString(ourTextBox.getText(), point.x, point.y);
 }
 public static void main (String[] args) {
  CTwoFrames cT = new CTwoFrames();
  cT.show();
 }
}
```

Понятие об Layout-ах

```
/* FlowLayout*/
import java.awt.*;
class Flow extends Frame {
 public Flow() {
  this.setSize(200, 100);
  String items[]= {"1", "2", "3", "4", "5", "6", "7", "8", "9"};
  Button but[]= new Button[10];
  setLayout (new FlowLayout (FlowLayout.LEFT));
  for (int i= 0; i < items.length; i++) {
   but[i]= new Button (items[i]);
   add(but[i]);
  }
  this.show ( );
 }
 public static void main (String[] args) {
  Flow f = new Flow();
 }
}
/*
```

Result: В прикладном окне менеджер расстановки разместил 10 кнопок слева направо сверху вниз. См. рисунок ниже.


```
*/
//
//-------------------------------------------------------------------------------------------------------
//
/* BorderLayout*/
import java.awt.*;
class Border extends Frame {
 public Border() {
  this.setSize(200, 100);
  String items[]= {"1", "2", "3", "4", "5"};
  String locs[]= {"North", "South", "East", "West", "Center"};
  Button but[]= new Button[10];
  setLayout(new BorderLayout ( ));
```

```
 for (int i= 0; i < 5; i++) {
  /*this.*/add (locs[i], new Button (items[i]));
 }
 this.show ( );
}
public static void main (String[] args) {
 Border b = new Border();
}
}
/*
```

Result: В прикладном окне менеджер расстановки разместил 5 кнопок в соответствии с указаниями. См. рисунок ниже.


```
*/
//
//-----------------------------------------------------------------------------------------------------------
//
/* GridLayout*/
import java.awt.*;
class Grid extends Frame {
 public Grid() {
  this.setSize(200, 100);
  String items[]= {"1", "2", "3", "4", "5", "6", "7", "8", "9"};
  Button but[]= new Button[10];
  setLayout(new GridLayout (2, 5));
  for (int i= 0; i < items.length; i++) {
   add (new Button (items[i]));}
   this.show();
  }
 public static void main (String[] args) {
  Grid g = new Grid();
 }
}
/*
```

Result: В прикладном окне менеджер расстановки разместил 10 кнопок в 2 ряда с не более 5 кнопок в ряду. См. рисунок ниже.


```
*/
```

Пакеты

Пакет представляет собой исходный файл, содержащий набор описаний классов и оператор package, указывающих имя пространство имён (имя пакета) для этих классов. При необходимости можно создать иерархию пакетов. При этом в операторе package нужно использовать составное имя пакета, в котором имя данного пакета отделяется от имени вышестоящего пакета с помощью точки. При использовании пакета необходимо применить оператор import, указав в нём имя требуемого пакта.

Пакет java.lang имеет особую значимость, поскольку в нём содержатсяосновные широко используемые классы, такие как Object, System, String, Thread, Math, классы-оболочки и др. В виду своей важности пакет java.lang автоматически импортируется во все программы.

Пакет java.util содержит классы и интерфейсы коллекций, позволяющих создать массивы, списки, словари, стеки, очереди, векторы, деревья и др. Особый интерес представляют и другие классы этого пакета, например, класс Random для получения случайных чисел, класс Observable и интерфейс Observer для реализации уведомлений.

Пакет java.io связан с операциями ввода-вывода с различных устройств; пакет java.net поддерживает работу с сетью; пакет java.applet импортируется в апплеты; объёмный пакет java.awt применяется при работе с окнами, включая не только оконные классы, классы для графики и текста, но и пакет java.awt.event для реализации событий.

Пример 1. Программа, где используются элементы GUI, события и потоки

```
import java.util.*;
import java.util.LinkedList;
import java.awt.*;
import java.awt.event.*;
public class Test {
 static int count;
 public static void main(String[] args){
  count = 0;
  Balls balls = new Balls();
 }
}
class Balls extends Frame implements Observer, ActionListener, ItemListener {
 private LinkedList LL = new LinkedList();
 private Color col;
 private Frame f;
 private Button b;
 private Choice c;
 private TextField tf;
 Balls(){
  this.addWindowListener(new WindowAdapter2());
  f = new Frame();
  f.setSize(new Dimension(300,100));
  f.setTitle("Контроль");
  f.setLayout(new GridLayout());
  f.addWindowListener(new WindowAdapter2());
  b = new Button("Пуск");
  b.setSize(new Dimension(10,40));
  b.setActionCommand("OK");
```

```
 b.addActionListener(this);
 f.add(b, new Point(20,20));
 c = new Choice();
 c.addItem("Синий");
 c.addItem("Зелёный");
 c.addItem("Красный");
 c.addItem("Чёрный");c.addItem("Жёлтый");
 c.addItemListener(this);
 f.add(c, new Point(60,20));
 tf = new TextField();
 f.add(tf);
 f.setVisible(true);
 this.setSize(500,200);
 this.setVisible(true);
 this.setLocation(100, 150);
}
public void update(Observable o, Object arg) {
 Ball ball = (Ball)arg;
 System.out.println ("x= " + ball.thr.getName() + ball.x);
 repaint();
}
public void paint (Graphics g) {
 if (!LL.isEmpty()){
  for (Object LL1 : LL) {
   Ball ball = (Ball) LL1;
   g.setColor(ball.col);
   g.drawOval(ball.x, ball.y, 20, 20);
  }
 }
}
public void itemStateChanged (ItemEvent iE) {}
public void actionPerformed (ActionEvent aE) {
String str = aE.getActionCommand();
if (str.equals ("OK")){
 switch (c.getSelectedIndex()) {
  case 0: col= Color.blue; break;
  case 1: col= Color.green; break;
  case 2: col= Color.red; break;
  case 3: col= Color.black; break;
  case 4: col= Color.yellow; break;
 }
 Ball ball= new Ball(col, this.tf.getText());
 LL.add(ball);
 ball.addObserver(this);
}
repaint();
 }
}
class Ball extends Observable implements Runnable {
 Thread thr;
 private boolean xplus;
 private boolean yplus;
```

```
int x; int y;
Color col;
public Ball (Color col, String text) {
 xplus = true; yplus = true;
 x = 0; y = 30;
 this.col = col;
 Test.count++;
 thr = new Thread(this,Test.count+":"+text+":");
 thr.start();
}
public void run(){
 while (true){
  if(x==475) xplus = false;
  if(x==-1) xplus = true;
  if(y==175) yplus = false;
  if(y==29) yplus = true;
  if(xplus) x++; else x--;
  if(yplus) y++; else y--;
  setChanged();notifyObservers (this);
  try{Thread.sleep (10);}
  catch (InterruptedException e){}
 }
}
}
class WindowAdapter2 extends WindowAdapter {
 public void windowClosing (WindowEvent wE) {System.exit (0);}
}
```

Практическая часть

1. В зависимости от варианта выполняется разработка приложения. Пример приложения похожего на то, которое нужно разработать см. в примере 1.
2. Разрабатываемое приложение должно состоять из двух окон: управляющего (УО) и дочернего, т.е. демонстрационного (ДО). В первом должны отображаться интерфейсные элементы (кнопки, текстовые поля, выпадающие списки, элементы выбора цвета), во втором в зависимости от настроек, сделанных в первом окне, должны передвигаться фигурки или простые объекты (ФиО).
3. При выставлении в УО некоторых настроек запуска очередной ФиО и нажатии на кнопку «Пуск» в ДО из левого верхнего угла в случайном направлении должен начать двигаться ФиО (**!!!** т.е. приращение координат «икс» и «игрек» ФиО может быть таким, что оно будет отличаться от траектории, совпадающей с биссектрисой угла вылета ФиО).
4. ФиО при движении в ДО должны отражаться от сторон окна по правилу: «*угол отражения равен углу падения*».
5. В УО должна быть предусмотрена возможность выбора уже запущенного ФиО и изменения его параметров (например, цвет, скорость).
6. Номер, который присваивается ФиО должен отображаться рядом с ним (ФИО), когда он (ФИО) перемещается в ДО.
7. Все ФиО должны быть пронумерованы, при чём, если предусмотрено изменение номера ФиО, то необходимо обеспечить уникальность номера каждого ФиО.
8. Приложение должно закрываться при закрывании любого из окон.

Варианты заданий

ФиО: фигуры – 1 (круг, овал, треугольник, квадрат, прямоугольник) или объекты – 2 (маленькая простая картинка или надпись, которая задаётся из УО).

Таблица 1. Задания на лабораторную работу №6

№	*Число ФиО* 1 – любое 2 – задано в коде УО	*ФиО* 1 – фигуры 2 – объекты	*Задание цвета текста и заливки ФиО* 1 – выпад. список назв. (пять и более); 2 – стандарт. элемент выбора цвета;	*Выбор запуска**емого** ФиО* 1 – выпад. список назв.; 2 – из текст. поля (круг, овал…);	*Задание начальной скорости ФиО* 1 – указанием в текст. поле; 2 – из выпадающего списка (шесть скоростей);	*Способ выбора запу**щенного** ФиО* 1 – из выпад. списка; 2 – из текст. поля, где вводится номер ФиО;	*Присвоение номера ФиО* 1 – авто; 2 – вручную из УО;	*Возможность смены номера ФиО из УО* 1 – нет 2 - да	*Регулировка скорости перемещения выбран-го ФиО* 1 – указанием в текст. поле; 2 – из выпадающего списка (пять скоростей);	*Изменения размера окна отображения ФиО* 1 – нет; 2 – да; (т.е. отражение ФиО в новых границах)
1.	1	1	2	2	1	2	2	2	2	2

Число в указанном варианте следует толковать как двоичное число, записанное единицами и двойками (где 0 это единица, а 1 это двойка), каждый разряд которого соответствует столбцу из таблицы 1

1	223	18	57	35	419	52	289	69	549	86	364
2	918	19	797	36	896	53	623	70	1010	87	67
3	76	20	928	37	809	54	256	71	205	88	430
4	366	21	850	38	307	55	909	72	147	89	375
5	115	22	354	39	174	56	609	73	551	90	547
6	631	23	56	40	598	57	737	74	118	91	139
7	506	24	517	41	139	58	386	75	621	92	621
8	95	25	606	42	149	59	964	76	900	93	700
9	303	26	583	43	316	60	124	77	264	94	142
10	650	27	733	44	667	61	944	78	174	95	914
11	787	28	911	45	160	62	642	79	191	96	68
12	556	29	821	46	1013	63	761	80	819	97	523
13	119	30	138	47	319	64	997	81	50	98	546
14	557	31	168	48	431	65	539	82	883	99	606
15	25	32	370	49	849	66	859	83	423	100	442
16	15	33	896	50	320	67	510	84	518	101	306
17	251	34	553	51	215	68	205	85	370		