МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра "ПОВТ и АС"

Рисование и обработка событий в пакетах AWT и Swing

Методические указания к лабораторной работе

по дисциплинам "Программирование", "Объектно-ориентированное программирование"

Ростов-на-Дону 20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

Рисование и обработка событий в пакетах AWT и Swing: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – 8 с.

В методической разработке рассматриваются вопросы использования элементов 2D-графики для отображения графиков гладких функций на заданном интервале значений аргумента из пакетов AWT и Swing, общая схема рисования и обработки событий, обработка событий от мыши, клавиатуры, таймеров. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов направлений 090304 "Программная инженерия", 020303 "Математическое обеспечение и администрирование информационных систем".

Ответственный редактор:

Издательский центр ДГТУ, 20

С самой первой версии языка Java среди стандартных библиотек (пакетов) присутствовал пакет AWT (java.awt), содержащий, помимо прочего, набор классов, представляющих элементы графического пользовательского интерфейса (Graphical User Interface - GUI). Фактически эти классы представляют собой оболочки вокруг обращений к компонентам пользовательского интерфейса, имеющимся в конкретной операционной системе (ОС) под управлением которой выполняется Java-приложение. С точки зрения операционной системы эти компоненты - это окна, способные реагировать на внешние воздействия (события). Если пользователь воздействует на эти окна с помощью, например, мыши или клавиатуры, то операционная система оповещает об этом соответствующие компоненты. Так как Java-программы являются кроссплатформенными, набор элементов пользовательского интерфейса должен быть такими, чтобы его поддержка имелась во всех операционных системах и/или оболочках для которых существует виртуальная машина Java (JVM). То есть этот набор - это общее подмножество элементов GUI разных ОС. Отсюда следует одно из ограничений AWT - наличие только самых простых элементов управления и невозможность произвольного расширения их набора. Элементы GUI, которые являются окнами конкретной операционной системы и которые тем самым напрямую взаимодействуют с ОС называют тяжеловесными. Элементы способные иметь произвольную форму, быть прозрачными и т.п. нужно создавать не как стандартные окна конкретной ОС. Такие элементы называют легковесными. Фактически они являются областями внутри окна некоторого компонента верхнего уровня и ОС о них ничего не знает. Тем самым ОС не генерирует события для таких компонентов. События связываются с тяжеловесными компонентами и уже они должны передавать эти сообщения легковесным компонентам, размещенным в их клиентской области. Начиная с версии 1.2 стандартной библиотекой для создания GUI стала библиотека Swing (пакет javax.swing) (а затем библиотека JavaFX). В ней используются легковесные компоненты, за исключением окон верхнего уровня. Там, где используются тяжеловесные компоненты, Swing обращается к компонентам AWT.

В пакете AWT помимо компонентов GUI определены и другие классы, которые используются и в библиотеке Swing. Класс Color представляет понятие композитного цвета. Обычно цвет пикселя формируется смешением значений его красной, зеленой и синей составляющих (RGB-цвет, R-red, G-

green, B-blue), кроме того, композитный цвет может содержать и альфа-канал, задающий степень прозрачности пикселя. По сути, значение цвета - это целое значение (типа int, то есть занимает 4 байта), в котором каждый байт, от младшего к старшему, содержит красную, зеленую, синюю составляющие и значение альфа-канала. Так как в один байт можно записать значение от 0 до 255 включительно, каждая составляющая может принимать одно из этих значений. В классе Color предопределены константы (статические) для разных цветов: WHITE, white, BLACK, black, GRAY, gray, LIGHT_GRAY, light_gray, DARK_GRAY, dark_gray, RED, red и т.д. (см. документацию). Конструкторы и другие методы:

Color(int r, int g, int b)

Color(int r, int g, int b, int alpha)

int getRed()

int getGreen()

int getBlue()

int getAlpha()

int getRGB()

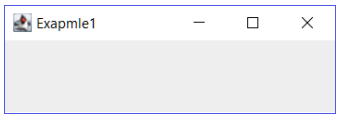
В AWT содержится класс Graphics для представления понятия графический контекст (см. ниже), а также стандартные интерфейсы и классы для системы обработки событий в пакете java.awt.event (см. ниже).

1. Компоненты GUI

Библиотека графического пользовательского интерфейса AWT предлагает использовать в качестве главного окна программы объект класса Frame. В Swing для этого используется наследник Frame класс JFrame. Вообще подавляющее большинство имен классов, представляющих компоненты GUI в Swing начинаются с буквы J. Frame/JFrame представляет окно с полосой заголовка на которой, помимо заголовка, размещаются также элементы управления для сворачивания окна, разворачивания его на весь экран и закрытия окна, рамкой, позволяющей изменять размеры окна и рабочей областью, которая является контейнером, то есть может содержать другие элементы пользовательского интерфейса. Не только окно верхнего уровня, но

и вложенные окна в свою очередь могут быть контейнерами, содержащими в себе какое-то количество других компонентов: кнопок, списков, вложенных окон (JPanel). JPanel представляет подчиненное окно без заголовка и элементов для минимизации и разворачивания окна. Объект JPanel размещается в рабочей области JFrame или внутри другого JPanel. Поверхность JPanel можно использовать для рисования. Итак, чтобы получить программу с графическим пользовательским интерфейсом нужно создать окно верхнего уровня. Например, создать объект класса JFrame. В классе JFrame есть конструктор с одним аргументом - строкой, содержимое этой строки отображается в заголовке окна. Размер окна задается с помощью метода setSize(int width, int height). Для того, чтобы при закрытии окна завершалась работа всей программы нужно вызвать метод setDefaultCloseOperation и передать ему значение JFrame.EXIT_ON_CLOSE. Если этого не сделать, то по умолчанию окно закрывается, но приложение не завершает свою работу.

Вновь созданное окно не отображается на экране автоматически. Нужно явно сделать его видимым с помощью вызова метода setVisible(true).




```
import java.awt.*;
import javax.swing.*;

public class Example1 {
	private JFrame frame;

	public Example1() {
		frame = new JFrame("Example1"); // главное окно с заголовком
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setVisible(true); /* изначально окно не отображается на экране */
	}

	public static void main(String[] args) {
		new Example1(); // все создается в конструкторе
	}

}
```

Приложение часто строится как класс-наследник JFrame, например,

```
public class Example1 extends JFrame {
```

```
    public Example1() {
        super("Example1");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        new Example1().setVisible(true);
    }

}
```

2. Рисование в окне

Рисование в окне (JFrame, JPanel) проводится в методе public void paint(Graphics g). Метод paint вызывается не только при запуске программы, но и во всех ситуациях, когда нужно перерисовать ее окно. Например, когда изменяется размер этого окна. Единственный аргумент метода - ссылка на графический контекст, абстракцию, применяемую для представления устройства, поддерживающего графический вывод. В общем случае, метод класса, ответственный за рисование, не должен знать с каким устройством он работает. Т.е. рисование отрезка прямой должно выглядеть в программе совершенно одинаково и для дисплея, и для принтера или плоттера. Более того, контекст может относиться не к дисплею в целом, но к области окна программы или к области, занятой компонентом, размещенным в окне. Все функции рисования реализованы как методы класса Graphics из библиотеки (пакета) java.awt, поэтому нарисовать что-либо можно только указав контекст (объект класса Graphics) устройства вывода графики. Например, отрезок прямой линии от точки (10, 20) до точки (100, 250), можно нарисовать, вызвав метод drawLine(10. 20, 100, 250) класса Graphics. По умолчанию при рисовании ось OY направлена из верхнего левого угла окна (координаты 0,0) вниз, а ось OX вправо. Ширину и высоту окна (JFrame, JPanel) в пикселях можно узнать с помощью методов int getWidht() и int getHeight() соответственно.

Графическое изображение (обычно хранящееся в JPEG или GIF файле) может быть представлено в программе с помощью объектов класса Image или его наследника класса BufferedImage из пакета java.awt.image. Чтобы прочитать изображение из файла и связать с ним объект BufferedImage можно использовать класс File из пакета java.io и статический метод read класса ImageIO из пакета javax.imageio. Если файл не будет найден возникнет исключение IOException. Путь к файлу с изображением удобно задавать относительно текущего для приложения каталога, путь к этому каталогу можно получить, используя системное свойство user.dir. Например, если

нужно прочитать файл a.jpg, расположенный в подкаталоге images текущего каталога приложения

```
import java.awt.*; // импортируем все имена из пакета  AWT
import java.awt.image.*;
import java.io.*;
import java.imageio.ImageIO; // импортируем только имя класса ImageIO
import javax.swing.*;

class Example2 extends JPanel {
	private BufferedImage im;
	public Example2() {
		String imFile = System.getProperty("user.dir") + "/images/a.jpg";
		try {
			im = ImageIO.read(new File(imFile));
		}
		catch(IOException ioe) {
			im = null;
		}
	}
	…
}
```

Класс BufferedImage позволяет работать с отдельными пикселями изображения. Для этого можно использовать методы
int getRGB(int x, int y) - возвращает цвет пикселя в координатах (x, y)
void setRGB(int x, int y, int pixel) - задает цвет пикселя в координатах (x, y)

Некоторые методы класса Graphics:
Color getColor() - возвращает текущее (заданное в графическом контексте) значение цвета (линий, заливки, текста)
void setColor(Color c) - устанавливает текущее значение цвета
Font getFont() - возвращает текущий шрифт (описание класса Font см. в документации Java)
void setFont(Font f) - устанавливает текущий шрифт
void drawLine(int x1, int y1, int x2, int y2) - рисует отрезок прямой от точки (x1, y1) до точки (x2, y2) текущим цветом
void drawString(String str, int x, int y) - выводит текст начиная с координат (x, y) текущими шрифтом и цветом
void drawPolyline(int[] x, int[] y, int count) - рисует ломанную линию, определяемую точками с координатами x, y

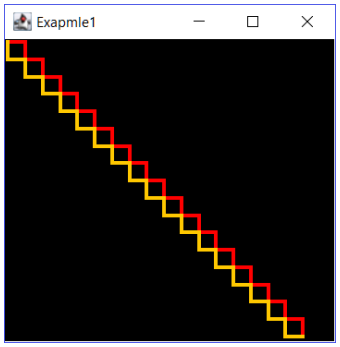
void drawPolygon(int[] x, int[] y, int count) - рисует замкнутую ломанную линию (полигон)
void fillPolygon(int[] x, int[] y, int count) - рисует закрашенный полигон
void drawPolygon(Polygon p) - рисует замкнутую ломанную линию, (описание класса Polygon см. в документации Java)
void fillPolygon(Polygon p) - рисует закрашенный полигон
void drawRect(int x, int y, int width, int height) - рисует прямоугольник, (x, y) - координаты левого верхнего угла, width, height - ширина и высота прямоугольника соответственно
void fillRect(int x, int y, int width, int height) - рисует закрашенный прямоугольник, (x, y) - координаты левого верхнего угла, width, height - ширина и высота прямоугольника соответственно
void drawOval(int x, int y, int width, int height) - рисует эллипс, размеры определяются прямоугольником в который мог бы быть вписан этот эллипс
void fillOval(int x, int y, int width, int height) - рисует закрашенный эллипс
void drawImage(Image im, int x, int y, ImageObserver io) - рисует графическое изображение полученное из фала (JPEG, GIF) или построенное в памяти. Левый верхний угол рисуется в координатах (x, y). ImageObserver позволяет управлять процессом отображения (рендеринга). При стандартном отображении можно указать в качестве ImageObserver ссылку на объект JFrame или JPanel, можно также задать пустую ссылку null
void drawImage(Image im, int x, int y, int width, int height, ImageObserver io) - рисует графическое изображение полученное из фала (JPEG, GIF) или построенное в памяти. Левый верхний угол рисуется в координатах (x, y), изображение масштабируется до размеров width и height по ширине и высоте соответственно

На самом деле давным-давно в AWT и Swing в качестве представителя графического контекста используется объект класса Graphics2D класса-наследника Graphics, но, для совместимости со старыми приложениями, ссылка на объект Graphics2D в методы paint, paintComponent и т.п. передается как ссылка на базовый класс Graphics. Graphics2D обладает большими возможностями чем Graphics. Чтобы получить доступ к ним, нужно привести ссылку на Graphics к ссылке на Graphics2D. Например, в Graphics толщина линии, которой рисуются отрезки прямой, эллипсы, прямоугольники т.п. имеет ширину в один пиксель. В Graphics2D можно использовать класс BasicStrok и метод setStrok, чтобы задать нужную ширину линии в пикселях.

Для легковесных компонентов метод paint вызывает последовательно методы прорисовки клиентского окна void paintComponent(Graphics g), метод прорисовки рамки void paintBorder(Graphics g), которая может быть у любого

легковесного компонента Swing (правда как элемент оформления, использовать ее для изменения размера окна нельзя) и метод прорисовки дочерних компонентов, размещенных в окне данного компонента void paintChildren(Graphics g). Поэтому, если мы рисуем что-то на поверхности легковесного компонента, например, JPanel, удобно переопределять не метод paint, а метод paintComponent. Перерисовку компонента в нужный момент можно инициировать вызвав метод repaint().

Пример приложения, в котором в главное окно добавляется панель, занимающая всю клиентскую область окна, и на поверхности панели отображается простенький рисунок.




```
import java.awt.*;
import javax.swing.*;

public class Example1 {
   private JFrame frame;
   private JPanel panel;

   public Example1 () {
      frame = new JFrame("Exapmle1");
      frame.setSize(300, 300);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      panel = new JPanel() { // внутренний класс-наследник JPanel
         @Override
         public void paintComponent(Graphics g) {
            int w = getWidth(), h = getHeight();
            g.setColor(Color.BLACK);
            g.fillRect(0, 0, w-1, h-1);
            Graphics2D g2 = (Graphics2D)g;
            g2.setStroke(new BasicStroke(3)); // толщина линий - 3 пикселя
            int step = 15;
            g.setColor(Color.RED);
            for(int i=3; i<w-step-20; i += step) {
               g.drawLine(i, i, i+step, i);
               g.drawLine(i+step, i, i+step, i+step);
            }
            g.setColor(Color.ORANGE);
            for(int i=3; i<w-step-20; i += step) {
               g.drawLine(i, i, i, i+step);
               g.drawLine(i, i+step, i+step, i+step);
```

```
            }
        }
    };
    frame.add(panel);
    frame.setVisible(true);
  }

  public static void main(String[] args) throws Exception {
    new Example1();
  }
}
```

При отображении графика функции на экране приходится проводить преобразование координат точек из системы координат, используемой в предметной области (физических координат) к системе координат окна, в котором выполняется рисование (оконным координатам). Например, предположим, что рисование выполняется в окне 100 x 200 пикселей. Тогда минимальное возможное значение x-координаты точки, видимой в окне, равно 0, а максимальное 99. Для y-координаты 0 и 199 соответственно. Координаты точек реальной задачи (не оконная система координат) могут принимать значения в любом другом диапазоне и, в общем случае не являются целыми числами и не обязательно положительные. Обозначим предельные значения оконных координат wndXMin, wndXMax, wndYMin и wndYMax. В примере выше их значения таковы: wndXMin = 0, wndYMin = 0, wndXMax = 99, wndYMax = 199. Пусть координаты конкретной точки реальной задачи x и y, а максимальные и минимальные возможные значения в этой системе координат: xMin, xMax, yMin и yMax. При переходе к оконной системе координат x и y должны перейти в wndX и wndY соответственно. Формулы преобразований таковы:

wndX = (x - xMin) / (xMax - xMin) * (wndXMax - wndXMin) + wndXMin
wndY = wndYMax - (y - yMin) / (yMax - yMin) * (wndYMax - wndYMin)

Далее нужно округлить полученные значения до ближайших целых и привести к типу int: (int)Math.round(…).

3. Обработка событий

С компонентами AWT и Swing могут быть связаны различные события. Например, если щелкнуть мышью по поверхности главного окна программы, возникнет событие, представленное объектом класса MouseEvent. Объект хранит информацию о событии, например, ссылку на источник

события (в данном случае это главное окно), координаты курсора мыши в момент щелчка, то, какой кнопкой мыши выполнен щелчок. Этот объект будет помещен в очередь событий приложения. В специальном потоке команд, работающем параллельно основному потоку (тому, в котором запущен метод main) выполняется специальная программа - обработчик событий. Компоненты программы, заинтересованные в получении сообщений о том, что событие произошло должны зарегистрироваться в источнике события. При этом они должны реализовать некоторый предопределенный метод (или методы). Собственно, этот метод и будет вызван, когда произойдет событие. Какой метод или какие методы нужно реализовать? Это определяется некоторым предопределенным интерфейсом. В AWT и Swing компоненты, желающие получать оповещения называются слушателями (Listeners). Например, есть некоторая панель (объект класса JPanel) и мы хотим реагировать на щелчки мышью в окне этого компонента. Здесь источником события от мыши будет выступать панель. Наш код, который выступает в роли слушателя должен реализовать предопределенный интерфейс MouseListener. В этом интерфейсе определено несколько методов, в том числе метод void mouseClicked(MouseEvent me), в случае щелчка мышкой будет вызываться именно этот метод. Чтобы реализовать интерфейс мы должны реализовать все описанные в нем методы, но, в данном случае, нас интересует только метод mouseClicked. Для таких случаев, то есть для всех интерфейсов слушателей в которых объявлено более одного метода, AWT предлагает классы-адаптеры, классы в которых реализованы все методы интерфейса, но эти реализации пустые, не содержат никаких действий. Теперь вместо реализации интерфейса, свой класс-слушатель мы можем сделать наследником класса адаптера и переопределить только тот метод, который нас интересует. Для интерфейса MouseListener класс-адаптер - это класс MouseAdapter. Классы-источники событий должны содержать методы addXXXListener(XXXEvent e), где XXX - имя события. Например, классы JFrame и JPanel содержат методы void addMouseListener(MouseListener ml) для регистрации слушателей событий от мыши связанных с объектами этих классов и методы void addKeyListener(KeyEvent ke) для регистрации слушателей событий от клавиатуры.

```
…
JPanel panel = new JPanel();
panel.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent me) {
                …
                panel.repaint();
```

```
        }
});
```
…
Теперь всякий раз, когда пользователь щелкает мышкой по области окна занятой panel будет вызываться метод mouseClicked который в данном примере будет инициировать перерисовку (вызов метода paint и, тем самым paintComponent) панели.

3.1. События от мыши и клавиатуры

С мышью связаны события MouseEvent, MouseWheelEvent и три слушателя: MouseListener, MouseMotionListener и MouseWheelListener.

В классе MouseEvent определены следующие основные методы:
int getX() - возвращает x-координату положения курсора мыши в окне компонента
int getY() - возвращает y-координату положения курсора мыши в окне компонента
int getButton() - возвращает целое значение, показывающее, какая кнопка мыши была нажата, это значение совпадает с одной из предопределенных констант: NOBUTTON (значение 0), BUTTON1 (1), BUTTON2 (2), BUTTON3 (3), если кнопок у мыши больше трех, в противном случае предопределенных констант нет и нужно использовать значения 4, 5 и т.д.
int getClickCount() - возвращает число кликов мышью

Класс MouseWheelEvent является наследником MouseEvent. Основной метод этого класса
int getWheelRotation() - возвращает число щелчков (кликов), характеризующих величину поворота колеса прокрутки

В интерфейсах объявлены следующие методы:
MouseListener
void mouseClicked(MouseEvent me) - возникает когда была нажата, а затем отпущена кнопка мыши
void mousePressed(MouseEvent me) - возникает когда была нажата кнопка мыши
void mouseReleased(MouseEvent me) - возникает когда была отпущена кнопка мыши
void mouseEntered(MouseEvent me) - возникает когда курсор мыши попадает в область, занятую компонентом

void mouseExited(MouseEvent me) - возникает когда курсор мыши покидает область, занятую компонентом

MouseMotionListener
void mouseMoved(MouseEvent me) - возникает, когда курсор мыши перемещается по области занятой компонентом, но ни одна кнопка мыши не нажата
void mouseDragged(MouseEvent me) - возникает, когда курсор мыши перемещается по области занятой компонентом, и нажата какая-то кнопка мыши

MouseWheelListener
void mouseWheelMoved(MouseWheelEvent mwe) - возникает, при вращении колеса прокрутки

Есть два класса-адаптера: MouseAdapter и MouseMotionAdapter с пустыми реализациями всех методов соответствующих интерфейсов.

С клавиатурой связаны событие KeyEvent и слушатель KeyListener.
В классе KeyEvent задано большое количество (статических) констант, например, VK_0, VK_1, … VK_9 - соответствуют символам, представляющим арабские цифры, VK_A, … VK_Z - буквам латинского алфавита, VK_PLUS, VK_ALT и т.д. специальным клавишам. Основные методы:
char getKeyChar() - возвращает символ (печатный), который соответствует нажатой на клавиатуре клавише
int getKeyCode() - возвращает код нажатой клавиши
Метод, полученный от родительского класса InputEvent
int getModifiers() - возвращает маску модификаторов, по которой можно узнать, были ли нажаты также управляющие клавиши Alt, Shift, Ctrl.

В интерфейсе KeyListener объявлены следующие методы:
void keyTyped(KeyEvent ke) - нажата клавиши, представляющая некоторый Unicode-символ. Введенный символ можно получить с помощью getKeyChar(), если вызвать getKeyCode() всегда вернется VK_UNDEFINED
void keyPressed(KeyEvent ke) - нажата клавиша, не обязательно представляющая некоторый Unicode-символ, это может быть, например, управляющий символ, чтобы получить код нужно вызвать getKeyCode()
void keyReleased(KeyEvent ke) - отпущена клавиша, не обязательно представляющая некоторый Unicode-символ, это может быть, например, управляющий символ, чтобы получить код нужно вызвать getKeyCode()

3.2. Таймеры

Таймеры можно использовать, например, для создания анимации. Классы таймеров определены в разных пакетах Java. В пакете Swing таймер генерирует событие ActionEvent через заданные интервалы времени. После запуска таймера первое событие ActionEvent будет сгенерировано после такого же интервала. Слушатели события должны реализовать интерфейс ActionListener. В этом интерфейсе объявлен один метод void actionPerformed(ActionEvent ae). При создании таймера нужно указать, помимо интервала генерации события ActionEvent, ссылку на объект класса, реализующего интерфейс ActionListener. После создания таймер нужно запустить вызвав метод start(). Можно остановить таймер с помощью метода stop().

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Example3 {
        private int x = 10, y = 40;
        private final int len = 20;
        private Timer t;

        public Example3() {
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setSize(300,200);
                t = new Timer( 200, new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent ae) {
                                int w = getWidth();
                                int h = getHeight();
                                x += 3;   y += 3;
                                if(x > w-1-len || y > h-1-len) {
                                        x = 10; y = 40;
                                }
                                repaint();
                        }
                });
                t.start();
        }
```

```
    @Override
    public void paint(Graphics g) {
        super.paint(g);
        g.fillRect(x, y, len, len);
    }

    public static void main(String[] args) {
        new Example3().setVisible(true);
    }
}
```

Во всех заданиях необходимо создавать не консольное, а оконное приложение.

ЗАДАНИЕ 1. Создайте приложение, отображающее в окне 300x300 пикселей график кривой f(x) = sin(x) на интервале x от -π до π.

ЗАДАНИЕ 2. Создайте класс Curve для отображения 2D-графика гладкой функции f(x) на заданном интервале изменения аргумента x. В классе должны быть предусмотрены два массива (или один двумерный массив) для хранения таблицы значений аргумента и функции, и метод setData, позволяющий пользователю класса задавать данные для конкретной функции и конкретного интервала по x. Необходимо предусмотреть методы для задания координат прямоугольной области, в которой должен отображаться график, и для отображения самого графика. Создайте приложение, отображающий в окне 300x300 пикселей график кривой f(x) = sin(x) на интервале x от 0 до 2*π с помощью объекта класса Curve.

ЗАДАНИЕ 3. Модифицируйте систему классов из задания 9 лабораторной работы "Классы в Java": Graph, Axis, Curve так, чтобы происходило реальное отображение графика в окне приложения.

ЗАДАНИЕ 4. Создайте приложение, в окне которого при щелчке мышью на месте курсора отображаются его координаты. Цвет отображения задается пользователем с помощью клавиатуры. Клавиши b, w, r, g, o задают цвет символов (b - black, w - white, r - red, g - green, o - orange).

ЗАДАНИЕ 5. Создайте приложение, загружающее и отображающее некоторое заданное в программе изображение из файла (JPEG или GIF). Изображение должно отображаться, занимая максимально возможную часть окна, подстраиваясь под изменение размера окна пользователем, но при этом отношение ширины изображения к его высоте должно оставаться неизменным (не должны меняться пропорции изображения). По щелчку мышью в области окна изображение в нем должно переворачиваться (поворачиваться на 180 градусов).

ЗАДАНИЕ 6. Эффект размытия (Blur) - каждая цветовая компонента пикселя (канал): красная, зеленая, синяя, заменяется на среднее значение соответствующих каналов соседних пикселей (и самого текущего пикселя). Например, можно взять область из 3x3 пикселей и рассмотреть центральный пиксель. Все остальные будут его ближайшими соседями. Находим среднее значение для каждого из их цветовых каналов и заменяем на эти значения цветовые каналы центрального пикселя. То же самое можно сделать для области 5x5 пикселей и т.д. Эффект преобразования в серые цвета - цветное изображение преобразуется в оттенки серого цвета. Для каждого пикселя находится среднее его значений по разным каналам: красному, зеленому и синему. Затем каждый канал получает это среднее значение. Измените программу из задания 5 так, чтобы с помощью клавиатуры можно было менять фильтры: вращение на 180 градусов - клавиша R или r, преобразование в оттенки серого (можно выполнить только один раз) - клавиша G или g, размытие - клавиша B или b (с учетом только ближайших соседей и самого пикселя). Собственно эффект возникает только после щелчка мышью по области окна.

ЗАДАНИЕ 7. Создайте приложение с анимацией. По щелчку мышью создается в произвольной позиции окна круг заданного цвета со случайным значением модуля скорости из некоторого диапазона. Можно задать для всех шариков направление скорости 45 градусов против часовой стрелки относительно горизонтальной оси. Шарики двигаются и отражаются от стенок. Столкновения не обрабатываются.

ЗАДАНИЕ 8. Создайте приложение "Бегущая строка". В программе задается массив символьных строк (сообщений) и в окне программы прокручивается одно из этих сообщение, за счет вывода этой строки так, как будто она движется по экрану слева-направо. Щелчок мышью

должен приводить к смене сообщения, то есть к выбору (случайным образом) другой строки из массива.

Литература

1. К.Арнольд, Дж.Гослинг, Д.Холмс "Язык программирования Java. 3-е изд.". – М.: Вильямс. – 2001, 624с.
2. И.Портянкин "Swing: эффективные пользовательские интерфейсы. 2-е изд.". – М.: Лори. – 2011, 592с.
3. М.Холл, Л.Браун "Программирование для Web. Библиотека профессионала". – М.: Вильямс. – 2002, 1264с.
4. П.Ноутон, Г.Шилдт "Java 2". – СПб.:BHV-Петербург. – 2001, 1072с.
5. Д.Бишоп "Эффективная работа: Java 2". – СПб.:Питер; К.:BHV. – 2002, 592с.

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01.	В набор	В печать
Объем 0,5 усл.п.л., уч.-изд.л.	Офсет.	Формат 60x84/16.
Бумага тип №3.	Заказ №	Тираж 140. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.