XXXX

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Программирование графики на Java. Рисование мышью

Методические указания к лабораторной работе

Рязань 2022

УДК 681.3

Программирование графики на Java. Рисование мышью: методические указания к лабораторной работе / Рязан. гос. радиотехн. ун-т.; сост. А.А. Митрошин, В.Г.Псоянц. – Рязань, 2022. – 16 с.

Содержат описание лабораторной работы, используемой в курсе «Компьютерная графика».

Предназначены для студентов дневной и заочной форм обучения направления «Информатика и вычислительная техника». Могут быть использованы для студентов других направлений, изучающих компьютерную графику или язык программирования Java.

Ил. --. Библиогр.: -- назв.

Программирование, компьютерная графика, Java.

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра САПР вычислительных средств Рязанского государственного радиотехнического университета (зав. кафедрой засл. деят. науки и техники РФ В.П.Корячко)

Программирование графики на Java. Рисование мышью

Составители: Митрошин Александр Александрович
Псоянц Владимир Грикорович

Редактор ___
Корректор ___
Подписано в печать 00.00.0000. Формат бумаги 60×84 1/16.
Бумага газетная. Печать трафаретная. Усл. печ. л. 1,0.
Уч-изд. л. 1,0. Тираж ___ экз. Заказ
Рязанский государственный радиотехнический университет.
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТУ.

События, связанные с мышью и их обработка

События мыши в компоненте возникают по одной из следующих причин:

- нажата кнопка мыши;
- отпущена кнопка мыши;
- щелчок кнопкой мыши (нажатие и отпускание не различаются);
- перемещение мыши;
- перемещение мыши с нажатой кнопкой;
- курсор мыши вошел в область компонента;
- курсор мыши покинул область компонента;
- повернуто колесико мыши.

Для обработки этих событий используются методы в интерфейсах MouseListener, MouseMotionListener и MouseWheelListener.

Методы интерфейса MouseListener

```
public interface MouseListener extends EventListener
```

Модификатор и тип	Метод и его описание
`void`	`mouseClicked(MouseEvent e)` Метод вызывается при щелчке по кнопке мыши (нажатии и отпускании) на компоненте
`void`	`mouseEntered(MouseEvent e)` Метод вызывается, когда мышь **входит** в границы компонента
`void`	`mouseExited(MouseEvent e)` Метод вызывается, когда мышь **выходит** за границы компонента
`void`	`mousePressed(MouseEvent e)` Метод вызывается, когда кнопка мыши **нажата** на компоненте
`void`	`mouseReleased(MouseEvent e)` Метод вызывается, когда кнопка мыши **отпущена** на компоненте

Методы интерфейса MouseMotionListener

```
public interface MouseMotionListener extends
EventListener
```

Модификатор	Метод и его описание

и тип	
void	mouseDragged(MouseEvent e) Метод вызывается, когда нажата кнопка мыши и указатель мыши перемещается по компоненту
void	mouseMoved(MouseEvent e) Метод вызывается, когда указатель мыши перемещается по компоненту, но никакая кнопка не нажата

Методы интерфейса MouseWheelListener

```
public interface MouseWheelListener extends
EventListener
```

Модификатор и тип	Метод и его описание
void	mouseWheelMoved(MouseWheelEvent e) Метод вызывается, когда колесико мыши повернуто

В качестве аргумента все методы интерфейсов MouseListener и MouseMotionListener получают объект класса MouseEvent, описывающий произошедшее событие.

Класс MouseEvent

Поля класса MouseEvent

Модификатор и тип	Поле и его описание
static int	BUTTON1 Определяет кнопку №1 (обычно левую); используется при вызове метода getButton()
static int	BUTTON2 Определяет кнопку №2 (обычно среднюю или одновременно двух кнопок на двухкнопочной мыши); используется при вызове метода getButton()
static int	BUTTON3 Определяет кнопку №3 (обычно правую); используется при вызове метода getButton()
static int	MOUSE_CLICKED Событие «Щелчок по кнопке мыши»
static int	MOUSE_DRAGGED Событие «Перемещение мыши с нажатой кнопкой»
static int	MOUSE_ENTERED Событие «Указатель мыши вошел в область компонента»

static int	MOUSE_EXITED Событие «Указатель мыши вышел из области компонента»
static int	MOUSE_FIRST Первое событие в последовательности событий
static int	MOUSE_LAST Последнее событие в последовательности событий
static int	MOUSE_MOVED Событие «Указатель мыши переместился»
static int	MOUSE_PRESSED Событие «Кнопка мыши нажата»
static int	MOUSE_RELEASED Событие «Кнопка мыши отпущена»
static int	MOUSE_WHEEL Событие «Колесико мыши повернуто»
static int	NOBUTTON Показывает отсутствие клавиш мыши; используется для getButton()

Методы класса MouseEvent

Модификатор и тип	Метод и его описание
int	getButton() Возвращает какая (какие) кнопки мыши изменили состояние
int	getClickCount() Возвращает количество кликов мыши, ассоциированное с событием
Point	getLocationOnScreen() Возвращает абсолютные значения координат указателя мыши во время наступления события
int	getModifiersEx() Возвращает расширенный модификатор маски для этого события
static String	getMouseModifiersText(int modifiers) Возвращает строку, описывающую клавиши клавиатуры и кнопки мыши, которые были нажаты во время наступления события, такие как Shift" или "Ctrl+Shift"
Point	getPoint() Возвращает позицию курсора мыши во время наступления события в системе координат компонента
int	getX() Возвращает координату x курсора мыши во время наступления события в системе координат компонента
int	getXOnScreen() Возвращает координату x курсора мыши во время наступления события в системе координат дисплея
int	getY() Возвращает координату y курсора мыши во время наступления

	события в системе координат компонента
`int`	`getYOnScreen()` Возвращает координату у курсора мыши во время наступления события в системе координат дисплея
`boolean`	`isPopupTrigger()` Возвращает является ли это событие мыши событием всплывающего меню для текущей платформы
`String`	`paramString()` Возвращает строку параметров, идентифицирующую событие
`void`	`translatePoint(int x, int y)` Преобразует координаты курсора мыши к новой позиции, добавляя определенные x (горизонтальное) и y (вертикальное) смещение

Класс MouseWheelEvent

```
public class MouseWheelEvent extends MouseEvent
```

Поля класса MouseWheelEvent

Модификатор и тип	Поле и его описание
`static int`	`WHEEL_BLOCK_SCROLL` Константа, представляющая скроллирование «блоком» (аналогично клавишам page-up и page-down)
`static int`	`WHEEL_UNIT_SCROLL` Константа, представляющая скроллирование «единицей» (аналогично клавишам со стрелками)

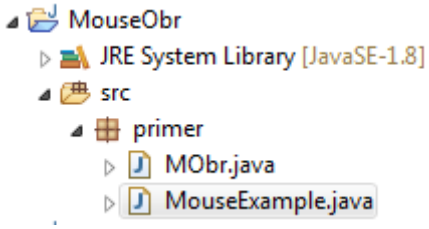
Методы класса MouseWheelEvent

Modifier and Type	Method and Description
`double`	`getPreciseWheelRotation()` Возвращает число «кликов» мыши колесика мыши при его вращении
`int`	`getScrollAmount()` Возвращает число единиц, на которое будет производиться скроллирование за один оборот колесика мыши
`int`	`getScrollType()` Возвращает тип скроллирования, который будет иметь место в ответ на это событие
`int`	`getUnitsToScroll()` Метод позволяет реализовать в общем случае MouseWheelListener – для скроллирования ScrollPane или JScrollPane
`int`	`getWheelRotation()` Возвращает количество «кликов» на которое колесико мыши повернуто. Если значение положительное, то колесико вращалось на себя, если

	отрицательное, то от себя
`String`	`paramString()` Возвращает строку параметров, идентифицирующих событие

Пример обработки событий мыши

Структура проекта



Класс MouseExample

```
package primer;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;

public class MouseExample {
	public static void main(String[] args) {
	//Создаем фрейм
		JFrame f = new JFrame("События мыши");
	//Получаем объект toolkit, который содержит много полезного
	//toolkit изучить самостоятельно!!!
		Toolkit tk=f.getToolkit();
	//Получаем размер экрана монитора как объект класса Dimension
		Dimension scrdim = tk.getScreenSize();
	//Получаем широту и высоту экрана монитора
		int scrX=scrdim.width;
		int scrY=scrdim.height;
	//Устанавливаем положение и размер фрейма
	//Если что-то не нравится, то установите другие
		f.setBounds(scrX/6, scrY/6, scrX/3, scrY/3);
	//Определяем реакцию на закрытие окна
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	//Добавляем к фрейму слушателей. Обработчики реализованы в классе MObr
		f.addMouseListener(new MObr());
		f.addMouseMotionListener(new MObr());
		f.addMouseWheelListener(new MObr());
		//делаем фрейм видимым
		f.setVisible(true);
	}
}
```

Класс MObr

```
package primer;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
```

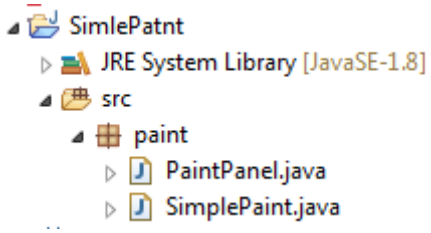
```
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;

public class MObr implements MouseListener, MouseMotionListener,
MouseWheelListener {
String st = new String("");
	@Override
	//Обработка вращения колесика мыши
	public void mouseWheelMoved(MouseWheelEvent mwe) {
		// Метод возвращает положительное значение, если колесико мыши
вращается на себя
		// и отрицательное, если вращение происходит от себя
		if (mwe.getWheelRotation() > 0)
				st = new String (" на себя" );
		if (mwe.getWheelRotation() < 0)
				st = new String (" от себя" );
		System.out.println("Колесико мыши повернуто" + st);
	}
	@Override
	//Обработка перемещения мыши с нажатой кнопкой мыши
	public void mouseDragged(MouseEvent me) {
		if ((me.getModifiers() & MouseEvent.BUTTON1_MASK) ==
MouseEvent.BUTTON1_MASK)
		System.out.println("Мышь движется с нажатой кнопкой - 1 (Левая)");
		if ((me.getModifiers() &
MouseEvent.BUTTON3_MASK)==MouseEvent.BUTTON3_MASK)
		System.out.println("Мышь движется с нажатой кнопкой - 3 (Правая)");
	}
	@Override
	//Обработка перемещения мыши
	public void mouseMoved(MouseEvent me) {
System.out.println("Перемешение мыши "+ "x= "+me.getX()+" y= "+me.getY());
	}
	@Override
	//Обработка клика и двойного клика
	public void mouseClicked(MouseEvent me) {
		System.out.println("Клик");
		if (me.getClickCount()==2) System.out.println("Двойной клик");
	}
	@Override
	//Обработка вхождения курсора мыши в область компонента,
	//в данном случае фрейма
	public void mouseEntered(MouseEvent me) {
		System.out.println("Мышь вошла в окно");
	}
	@Override
	//Лбработка выхода из пределов компонента
	public void mouseExited(MouseEvent me) {
		System.out.println("Мышь вышла из окна");
	}
	@Override
	//Обработка нажатия кнопки мыши
	public void mousePressed(MouseEvent me) {
		//System.out.println(me.getButton());
	System.out.println("Мышь нажата. Номер кнопки - " + me.getButton());
	}
	@Override
	//Обработка отпускания кнопки мыши
```

```
	public void mouseReleased(MouseEvent me) {
		System.out.println("Мышь отпущена");
	}
}
```

Пример простого графического редактора

Структура проекта



Класс SimplePaint

```
package paint;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;

public class SimplePaint extends JFrame {
	private static final long serialVersionUID = 1L;
	//Конструктор класса
	public SimplePaint(String s) {
		//Вызываем конструктор суперкласса, то есть класса JFrame
		//и передаем в него строку заголовка окна
		super(s);
		//Запрещаем менять размеры окна
		this.setResizable(false);
		//Устанавливаем размеры окна
		this.setSize(500, 500);
		//Создаем объект класса PaintPanel, который описан ниже
		PaintPanel panel = new PaintPanel(this, 800, 800);
		//Добавляем к созданному объекту обработчики событий
		//В нашем случае этот объект сам будет обрабатывать свои события
		panel.addMouseListener(panel);
		panel.addMouseMotionListener(panel);
		//Создаем скроллируемую панел, чтобы посмотреть, как это делается
		//Скроллировать эта панель будет панель для рисования panel, её и
		//передаем в конструктор JScrollPane
		JScrollPane pane = new JScrollPane(panel);
		//Определяем, чтобы вертикальная и горизонтальная полосы прокрутки
		// панели показывались всегда
pane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
pane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
		//Добавляем панель для рисования в центральную область нашего окна
		//this - ссылка на самого себя, то есть в нашем случае объект
		//класса SimplePaint
		this.add(pane, BorderLayout.CENTER);
		//Создаем новую панель
		JPanel p = new JPanel();
```

```
            //Добавляем эту панель в нижнюю часть нашего (южную) окна
            this.add(p, BorderLayout.SOUTH);
            //Создаем несколько кнопок, каждую из которых добавляем на панель
            //Определяем обработчика событий для каждой кнопки
            JButton b1 = new JButton("Красный");
            p.add(b1);
            b1.addActionListener(panel);
            JButton b2 = new JButton("Зеленый");
            p.add(b2);
            b2.addActionListener(panel);
            JButton b3 = new JButton("Синий");
            p.add(b3);
            b3.addActionListener(panel);
            JButton b4 = new JButton("Черный");
            p.add(b4);
            b4.addActionListener(panel);
            JButton b5 = new JButton("Очистить");
            p.add(b5);
            b5.addActionListener(panel);
            //Определяем действие при закрытии окна
            this.setDefaultCloseOperation(EXIT_ON_CLOSE);
            //Делаем окно видимым
            this.setVisible(true);
      }

      public static void main(String[] args) {
      //Создаем окно как безымянный объект, потому что имя его нам не нужно
            new SimplePaint("Очень простой редактор");
      }
}
```

Класс PaintPanel

```
package paint;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import javax.swing.JFrame;
import javax.swing.JPanel;

//Класс, расширяющий JPanel и реализующий интерфейсы
//ActionListener - слушатель событий лействия
//MouseListener - слушатель событий мыши
//MouseMotionListener - слушатель событий перемещения мыши
public class PaintPanel extends JPanel implements ActionListener, MouseListener,
MouseMotionListener {
      private static final long serialVersionUID = 1L;
      private float wl = 5.0F;
      protected int lastX, lastY, w, h;
      protected Color curColor = Color.BLACK;
      protected JFrame f;
```

```
	//Конструктор. Принимает в качестве параметров фрейм, на котором будет
размещена
	//панель и размеры панели
	public PaintPanel(JFrame frame, int width, int height) {
		super();
		f = frame;
		w = width;
		h = height;
	}
	//Обработчик события перемещения мыши с нажатой кнопкой
	@Override
	public void mouseDragged(MouseEvent me) {
		//Если при перемещении нажата левая кнопка мыши
		if ((me.getModifiers() & MouseEvent.BUTTON1_MASK) ==
MouseEvent.BUTTON1_MASK) {
		//С поммощью вызова метода this.getGraphics() получаем графический
контекст нашей панели
		//и приводим его к Graphics2D
		Graphics2D g2 = (Graphics2D)this.getGraphics();
		//Устанавливаем текущую ширина штриха (Stroke) в 5 пикселей
		g2.setStroke(new BasicStroke(wl));
		//Устанавливаем текущий цвет рисования
		g2.setColor(curColor);
		//Рисуем текущим штрихом и цветом прямую линию от предыдущего
		//положения мыши до текущего
		g2.drawLine(lastX, lastY, me.getX(), me.getY());
		//Телаем текущее положение мыши предыдущим
		lastX = me.getX();
		lastY = me.getY();
		}
	}
	//Это событие не обработано
	@Override
	public void mouseMoved(MouseEvent e) {}
	//Это событие не обработано
	@Override
	public void mouseClicked(MouseEvent arg0) {}
	//Это событие не обработано
	@Override
	public void mouseEntered(MouseEvent arg0) {}
	//Это событие не обработано
	@Override
	public void mouseExited(MouseEvent arg0) {}

	//Обработчик события нажания мыши
	@Override
	public void mousePressed(MouseEvent me) {
		//Если нажата левая кнопка мыши
		if ((me.getModifiers() & MouseEvent.BUTTON1_MASK) ==
MouseEvent.BUTTON1_MASK) {
		//устанавливаем предыдущие координаты мыши в текущие
		lastX = me.getX();
		lastY = me.getY();
		}
	}
	//Это событие не обработано
	@Override
```

```
	public void mouseReleased(MouseEvent arg0) {}

	//Обработчик события действия. Применяется здесь для
	//обработки нажатий на кнопки
	@Override
	public void actionPerformed(ActionEvent event) {
		String s = event.getActionCommand();
		//Если нажата кнопка "Очистить", то очистить панель рисования
		if (s.equals("Очистить")) this.repaint();
		//Если нажата кнопка "Красный", то установить текущий цвет
рисования в красный
		//Остальные кнопки аналогично
		else if (s.equals("Красный")) curColor = Color.RED;
		else if (s.equals("Зеленый")) curColor = Color.GREEN;
		else if (s.equals("Синий")) curColor = Color.BLUE;
		else if (s.equals("Черный")) curColor = Color.BLACK;
	}
}
```

Порядок выполнения работы

1. Изучите теоретический материал.
2. Выполните задание.
3. Ответьте на контрольные вопросы.

Варианты заданий

1. Измените работу простейшего графического редактора таким образом, чтобы вращение колесика мыши приводило к изменению размера пера, которым производится рисование.

Библиографический список

1. Хортон А. Java 2. – М.: Лори, 2008.

2. Хабибуллин И. Java 7. – СПб.: БХВ-Петербург, 2012.

Питер, 2002.