ЛАБОРАТОРНАЯ РАБОТА № 7-8
РАЗРАБОТКА ПРИЛОЖЕНИЯ В JAVA НА ОСНОВЕ МОДЕЛИ UML

1 Цель занятия

Сформировать практические навыки по использованию событий языка программирования Java с использованием программы Eclipse.

2 Общие теоретические сведения

2.1 Обработка событий в Java

Современный подход к обработке событий основан на модели делегирования событий (delegation event model), определяющей стандартные и согласованные механизмы для создания и обработки событий. Его концепция проста: источник извещает о событии одного или несколько слушателей (listener). В этой схеме слушатель просто ожидает до тех пор, пока не получит извещение о событии. Как только извещение о событии получено, слушатель обрабатывает его и возвращает управление.

В модели делегирования событие — это объект, описывающий изменение состояния источника. Он может быть создан в результате взаимодействия пользователя с элементом графического интерфейса. К событиям приводят такие действия, как щелчок на экранной кнопке, ввод символа с клавиатуры, выбор элемента в списке и щелчок кнопкой мыши. Многие другие пользовательские операции также могут служить подобными примерами.

События могут также происходить и не в результате прямого взаимодействия с пользовательским интерфейсом. Например, событие может произойти по истечении времени таймера в результате превышения счетчиком некоторого значения, программного или аппаратного сбоя либо завершения некоторой операции.

Источник (source) — это объект, извещающий о событии. Событие происходит при изменении внутреннего состояния объекта некоторым образом. Источники могут извещать о событиях нескольких типов.

Источник должен регистрировать слушателей, чтобы они получали извещение о событиях определенного рода. Каждый тип события имеет собственный метод регистрации. Вот его общая форма.

```
public void addТипListener(ТипListener el)
```

Здесь *Тип* — имя события, а *el* — ссылка на слушателя событий.

Интерфейсы	Обработчики события
ActionListener	actionPerformed(ActionEvent e)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)
ComponentListener	componentResized(ComponentEvent e)

Интерфейсы	Обработчики события
	componentMoved(ComponentEvent e) componentShown(ComponentEvent e) componentHidden(ComponentEvent e)
ContainerListener	componentAdded(ContainerEvent e) componentRemoved(ContainerEvent e)
FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)
ItemListener	itemStateChanged(ItemEvent e)
KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)
MouseListener	mouseClicked(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e)
MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
TextListener	textValueChanged(TextEvent e)
WindowListener	windowOpened(WindowEvent e) windowClosing(WindowEvent e) windowClosed(WindowEvent e) windowIconified(WindowEvent e) windowDeiconified(WindowEvent e) windowActivated(WindowEvent e)

Событие, которое генерируется в случае возникновения определенной ситуации и затем передается зарегистрированному блоку прослушивания для обработки, – это объект класса событий. В корне иерархии классов событий находится суперкласс EventObject из пакета java.util. Этот класс содержит два метода: getSource(), возвращающий источник событий, и toString(), возвращающий строчный эквивалент события. Абстрактный класс AWTEvent из пакета java.awt является суперклассом всех AWT-событий, связанных с компонентами. Метод getID() определяет тип события, возникающего вследствие действий пользователя в визуальном приложении. Ниже приведены некоторые из классов событий, производных от AWTEvent, и расположенные в пакете java.awt.event:

ActionEvent – генерируется: при нажатии кнопки; двойном щелчке клавишей мыши по элементам списка; при выборе пункта меню;

AdjustmentEvent – генерируется при изменении полосы прокрутки;

ComponentEvent – генерируется, если компонент скрыт, перемещен, изменен в размере или становится видимым;

FocusEvent – генерируется, если компонент получает или теряет фокус ввода;

TextEvent – генерируется при изменении текстового поля;

ItemEvent – генерируется при выборе элемента из списка.

2.2 Пример интерфейса MouseListener и обработки событий от мыши

Мы рассмотрим технологию написания слушателей на примере слушателей событий мыши.

Допустим есть окно:

```
SimpleWindow(){
  super("Окно с кнопкой");
  setDefaultCloseOperation(EXIT_ON_CLOSE);
  JButton button = new JButton("Кнопка", new ImageIcon("1.gif"));
  button.setMargin(new Insets(0, 10, 20, 30));
  button.setVerticalTextPosition(SwingConstants.TOP);
button.setHorizontalTextPosition(SwingConstants.LEFT);
  getContentPane().add(button); pack();
}
```

Слушатель событий от мыши должен реализовать интерфейс MouseListener. В этом интерфейсе перечислены следующие методы:

– public void mouseClicked(MouseEvent event) — выполнен щелчок мышкой на наблюдаемом объекте

– public void mouseEntered(MouseEvent event) — курсор мыши вошел в область наблюдаемого объекта

– public void mouseExited(MouseEvent event) — курсор мыши вышел из области наблюдаемого объекта

– public void mousePressed(MouseEvent event) — кнопка мыши нажата в момент, когда курсор находится над наблюдаемым объектом

– public void mouseReleased(MouseEvent event) — кнопка мыши отпущена в момент, когда курсор находится над наблюдаемым объектом

Чтобы обработать нажатие на кнопку, требуется описать класс, реализующий интерфейс MouseListener, причем метод mouseClicked() должен содержать обработчик события. Далее необходимо создать объект этого класса и зарегистрировать его как слушателя интересующей нас кнопки. Для регистрации слушателя используется метод addMouseListener(MouseListener listener).

Опишем класс слушателя в пределах класса окна SimpleWindow, после конструктора. Обработчик события будет проверять, ввел ли пользователь логин «Иван» (пароль проверять не будем) и выводить сообщение об успехе или неуспехе входа в систему:

```
class MouseL implements MouseListener {
  public void mouseClicked(MouseEvent event) {
    if (loginField.getText().equals("Иван"))
     JOptionPane.showMessageDialog(null, "Вход выполнен");
```

```
        else JOptionPane.showMessageDialog(null, "Вход НЕ выполнен");
       }
     public void mouseEntered(MouseEvent event) {}
     public void mouseExited(MouseEvent event) {}
     public void mousePressed(MouseEvent event) {}
     public void mouseReleased(MouseEvent event) {}
     }
```

Мы сделали слушателя вложенным классом класса SimpleWindow, чтобы он мог легко получить доступ к его внутренним полям loginField и passwordField. Кроме того, хотя реально мы обрабатываем только одно из пяти возможных событий мыши, описывать пришлось все пять методов (четыре имеют пустую реализацию). Дело в том, что в противном случае класс пришлось бы объявить абстрактным (ведь он унаследовал от интерфейса пустые заголовки методов) и мы не смогли бы создать объект этого класса. А мы должны создать объект слушателя и прикрепить его к кнопке. Для этого в код конструктора SimpleWindow() необходимо добавить команду:

```
ok.addMouseListener(new MouseL());
Это можно сделать сразу после команды:
JButton ok = new JButton("OK");
```

Чтобы кнопка ok обрела слушателя, который будет обрабатывать нажатие на нее, нам понадобилось описать новый (вложенный) класс.

Программа стала выглядеть загроможденной главным образом из-за того, что помимо полезного для нас метода mouseClicked() нам пришлось определять пустые реализации всех остальных, не нужных методов. В принципе, этого можно избежать.

Класс MouseAdapter реализует интерфейс MouseListener, определяя пустые реализации для каждого из его методов. Можно унаследовать своего слушателя от этого класса и переопределить те методы, которые нам нужны.

В результате предыдущее описание слушателя будет выглядеть более компактно:

```
ok.addMouseListener( new MouseAdapter()
{
   public void mouseClicked(MouseEvent event) {
   if (loginField.getText().equals("Иван"))
           JOptionPane.showMessageDialog(null, "Вход выполнен");
  else JOptionPane.showMessageDialog(null, "Вход НЕ выполнен");
  }
});.
```

2.3 Пример реализации интерфейса MouseListener и события MouseEvent

/ события нажатия клавиши мыши: MyRect.java */*

```
package chapt12;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
public class MyRect extends JApplet {
private Rectangle rect =
new Rectangle(20, 20, 100, 60);
private class AppletMouseListener//блок обработки событий
implements MouseListener {
/* реализация всех пяти методов интерфейса MouseListener */
public void mouseClicked(MouseEvent me) {
  int x = me.getX();
  int y = me.getY();
  if (rect.contains(x, y)) {
     showStatus(
     "клик в синем прямоугольнике");
  } else {
     showStatus("клик в белом фоне");
  }
}
// реализация остальных методов интрефейса пустая
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e) {}
public void mousePressed(MouseEvent e) {}
public void mouseReleased(MouseEvent e) {}
}
public void init() {
  setBackground(Color.WHITE);
  /* регистрация блока прослушивания */
  addMouseListener(new AppletMouseListener());
}
public void paint(Graphics g) {
  g.setColor(Color.BLUE);
  g.fillRect(rect.x, rect.y,
  rect.width, rect.height);
}
}
```

2.4 Работа с файлами

Класс File, определенный в пакете java.io, не работает напрямую с потоками. Его задачей является управление информацией о файлах и каталогах. Хотя на уровне операционной системы файлы и каталоги отличаются, но в Java они описываются одним классом File.

В зависимости от того, что должен представлять объект File - файл или каталог, мы можем использовать один из конструкторов для создания объекта:

```
File(String путь_к_каталогу)
File(String путь_к_каталогу, String имя_файла)
File(File каталог, String имя_файла)
```

Например:

```
// создаем объект File для каталога
File dir1 = new File("C://SomeDir");
// создаем объекты для файлов, которые находятся в каталоге
File file1 = new File("C://SomeDir", "Hello.txt");
File file2 = new File(dir1, "Hello2.txt");
```

Класс File имеет ряд методов, которые позволяют управлять файлами и каталогами. Рассмотрим некоторые из них:

– boolean createNewFile(): создает новый файл по пути, который передан в конструктор. В случае удачного создания возвращает true, иначе false

– boolean delete(): удаляет каталог или файл по пути, который передан в конструктор. При удачном удалении возвращает true.

– boolean exists(): проверяет, существует ли по указанному в конструкторе пути файл или каталог. И если файл или каталог существует, то возвращает true, иначе возвращает false

– String getAbsolutePath(): возвращает абсолютный путь для пути, переданного в конструктор объекта

– String getName(): возвращает краткое имя файла или каталога

– String getParent(): возвращает имя родительского каталога

– boolean isDirectory(): возвращает значение true, если по указанному пути располагается каталог

– boolean isFile(): возвращает значение true, если по указанному пути находится файл

– boolean isHidden(): возвращает значение true, если каталог или файл являются скрытыми

– long length(): возвращает размер файла в байтах

– long lastModified(): возвращает время последнего изменения файла или каталога. Значение представляет количество миллисекунд, прошедших с начала эпохи Unix

– String[] list(): возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

– File[] listFiles(): возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

– boolean mkdir(): создает новый каталог и при удачном создании возвращает значение true

– boolean renameTo(File dest): переименовывает файл или каталог

Работа с каталогами

Если объект File представляет каталог, то его метод isDirectory() возвращает true. И поэтому мы можем получить его содержимое - вложенные подкаталоги и файлы с помощью методов list() и listFiles(). Получим все подкаталоги и файлы в определенном каталоге:

```
import java.io.File;
public class Program {
    public static void main(String[] args) {
        // определяем объект для каталога
        File dir = new File("C://SomeDir");
        // если объект представляет каталог
        if(dir.isDirectory())
        {
            // получаем все вложенные объекты в каталоге
            for(File item : dir.listFiles()){
                if(item.isDirectory()){
                    System.out.println(item.getName() + "  \t folder");
                }
                else{
                    System.out.println(item.getName() + "\t file");
                }
             }
        }
    }
}
```

Теперь выполним еще ряд операций с каталогами, как удаление, переименование и создание:

```
import java.io.File;
 public class Program {
     public static void main(String[] args) {
                // определяем объект для каталога
        File dir = new File("C://SomeDir//NewDir");
        boolean created = dir.mkdir();
        if(created)
            System.out.println("Folder has been created");
        // переименуем каталог
        File newDir = new File("C://SomeDir//NewDirRenamed");
        dir.renameTo(newDir);
        // удалим каталог
        boolean deleted = newDir.delete();
        if(deleted)
            System.out.println("Folder has been deleted");
    }
```

```
}
```

Работа с конкретными файлами

Работа с файлами аналогична работе с каталога. Например, получим данные по одному из файлов и создадим еще один файл:

```
import java.io.File;
import java.io.IOException;
 public class Program {
     public static void main(String[] args) {
             // определяем объект для каталога
        File myFile = new File("C://SomeDir//notes.txt");
        System.out.println("File name: " + myFile.getName());
        System.out.println("Parent folder: " + myFile.getParent());
        if(myFile.exists())
            System.out.println("File exists");
        else
            System.out.println("File not found");

        System.out.println("File size: " + myFile.length());
        if(myFile.canRead())
            System.out.println("File can be read");
        else
            System.out.println("File can not be read");

        if(myFile.canWrite())
            System.out.println("File can be written");
        else
            System.out.println("File can not be written");

        // создадим новый файл
        File newFile = new File("C://SomeDir//MyFile");
        try
        {
            boolean created = newFile.createNewFile();
            if(created)
                System.out.println("File has been created");
        }
        catch(IOException ex){

            System.out.println(ex.getMessage());
        }
    }
}
```

При создании нового файла метод createNewFile() в случае неудачи выбрасывает исключение IOException, поэтому нам надо его отлавливать, например, в блоке try...catch, как делается в примере выше.

2.5 Решение кубических уравнений с рациональными корнями.

Решение подбором

Начнем с простейшего случая, когда x=0 является корнем кубического уравнения $Ax^3+Bx^2+Cx+D=0$.

В этом случае свободный член D равен нулю, то есть уравнение имеет вид $Ax^3+Bx^2+Cx=0$.

Если вынести x за скобки, то в скобках останется квадратный трехчлен, корни которого легко найти либо через дискриминант, либо по теореме Виета $x\left(Ax^2+Bx+C\right)=0$.

Если коэффициенты кубического уравнения $Ax^3+Bx^2+Cx+D=0$ являются целыми числами, то уравнение может иметь рациональные корни.

При $A\neq 1$, домножим обе части уравнения на A^2 и проведем замену переменных y = Ax:

$$Ax^3+Bx^2+Cx+D=0$$
$$A^3\cdot x^3+B\cdot A^2\cdot x^2+C\cdot A\cdot A\cdot x+D\cdot A^2=0$$
$$y=A\cdot x \quad \Rightarrow$$
$$y^3+B\cdot y^2+C\cdot A\cdot y+D\cdot A^2=0$$

Пришли к приведенному кубическому уравнению. Оно может иметь целые корни, которые являются делителями свободного члена. Так что выписываем все делители и начинаем их подставлять в полученное уравнение до получения тождественного равенства. Тот делитель y_1, при котором тождество получено, является корнем уравнения. Следовательно, корнем исходного уравнения является $x_1=\frac{y_1}{A}$.

Далее делим многочлен Ax^3+Bx^2+Cx+D на $x-x_1$ и находим корни полученного квадратного трехчлена.

Решение кубических уравнений по формуле Кардано

В общем случае, корни кубического уравнения находятся по формуле Кардано.

Для кубического уравнения $A_0x^3+A_1x^2+A_2x+A_3=0$ находятся значения $B_1=\frac{A_1}{A_0}$, $B_2=\frac{A_2}{A_0}$, $B_3=\frac{A_3}{A_0}$. Далее находим $p=-\frac{B_1^2}{3}+B_2$ и $q=\frac{2B_1^3}{27}-\frac{B_1B_2}{3}+B_3$.

Подставляем полученные p и q в формулу Кардано:

$$y=\sqrt[3]{-\frac{q}{2}+\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}+\sqrt[3]{-\frac{q}{2}-\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}$$

Значения кубических корней следует брать такими, чтобы их произведение было равно $-\frac{p}{3}$. В итоге, находим корни исходного уравнения по формуле $x = y - \frac{B_1}{3}$.

3 Задачи для самостоятельного решения студентами

3.1 Варианты на лабораторную работу

В ходе лабораторной работы нужно разработать программу, обеспечивающую работу с экранной формой. Вариант на текущее занятие определяет преподаватель.

3.2 Порядок выполнения лабораторной работы

На 7-й лабораторной работе

1. Разработать форму со следующими параметрами:

Для вариантов 1, 5: текстового редактора, содержащее на 3-х отдельных панелях 1) поле ввода текста (350 на 600) с выравниванием заполнением поля (кроме панели с заданием); 2) меню с 12 кнопками, названными по операциям редактирования с выравниванием по верху; 3) строки состояния с надписями, содержащими 4 параметра с названиями: «Число символов», «Число слов», «Число строк», «Число знаков препинания» с выравниванием снизу.

Для вариантов 2, 6: графического редактора, содержащее на 3-х отдельных панелях 1) поле ввода графики серого цвета (500 на 700) с выравниванием заполнением поля (кроме панели с заданием); 2) меню с 14 кнопками, названными по операциям редактирования с выравниванием по верху; 3) строки состояния с надписями, содержащими 3 параметра с названиями: «Число фигур», «Число цветов», «Время с загрузки (в минутах)» с выравниванием снизу.

Для вариантов 3, 7: анкеты регистрации на сайте интересов, содержащей на 4-х отдельных панелях 1) надпись с названием анкеты с выравниванием сверху; 2) поля ввода полей регистрации (10 полей ввода с 10 уточняющими надписями связанными с темой 2-й лабораторной работы) с выравниванием заполнением (кроме панели с заданием); 3) 4 кнопки: «Сохранить», «Вызвать ещё поля», «Очистить», «Выход» с выравниванием снизу; 4) строка состояния с 2-мя надписями: «Заполнено полей», «Время начала ввода» с выравниванием снизу.

Для варианта 4: анкеты заказа обеда, содержащей на 4-х отдельных панелях 1) надпись с названием ресторана с выравниванием сверху; 2) поля ввода полей заказа (8 полей ввода с 8 уточняющими надписями связанными с темой заказа еды) с выравниванием заполнением (кроме панели с заданием); 3) 5 кнопок: «Сохранить заказ», «Просмотреть окно корзины», «Очистить поля», «Выход с оплатой», «Выход без сохранения» с выравниванием снизу; 4) строка состояния с 3-мя надписями: «Корзина заполнена?», «Стоимость заказа», «Номер заказа» с выравниванием снизу.

Для варианта 8: анкеты заказа аксессуаров, содержащей на 4-х отдельных панелях 1) надпись с названием магазина с выравниванием сверху и по центру; 2) поля ввода полей заказа (7 полей ввода с 7 уточняющими надписями связанными с темой заказа аксессуаров) с выравниванием заполнением (кроме панели с заданием); 3) 5 кнопок: «Сохранить заказ», «Просмотреть окно корзины», «Инвертировать цвет заливки полей», «Выход с оплатой», «Выход без сохранения» с выравниванием снизу; 4) строка состояния с 3-мя надписями: «Корзина заполнена?», «Стоимость заказа», «Номер заказа» с выравниванием снизу и слева.

Для вариантов 9, 10: формы ввода результатов эксперимента, содержащее на 3-х отдельных панелях 1) 15 надписей к полям, 12 числовых полей с результатами расчётов, числом итераций расчётов, уровнями моделирования (придумать осмысленные названия), 1 строковое поле с описанием формулы расчётов, 1 поле времени даты с временем-датой моделирования, 1 строковое поле с ФИО исследователей с выравниванием заполнением поля (кроме панели с заданием); 2) меню с 7 кнопками, названными по операциям редактирования («Запуск модели», «Останов», «Задание числа итераций», «Задание уровня», «Задание формулы», «Выбор метода», «Выход») с выравниванием по верху; 3) строки состояния с надписями, содержащими 4 параметра с названиями: «Время моделирования (в сек.)», «Число результатов», «Число итераций», «Номер уровня» с выравниванием снизу.

В заголовке окна должны выводиться фамилии студентов, выполняющих работу, текущая дата и номер варианта.

2. Разработать вторую форму, содержащую анализ данных, введённых в форму. На форме должно быть 2 поля с поясняющими надписями, вычисляющих следующие данные:

– число заполненных полей;

– количество ошибок при заполнении полей.

Кроме того, на форме должны быть предусмотрено ещё 3 подписанных поля, в которых должны быть указаны дополнительные аналитические характеристики (число чего-либо, время чего-либо и т.д.), связанные с особенностями темы по варианту.

3. В MS Word сделать отчёт, в котором показать каким образом разработанные классы, состояния элементов и последовательность выполнения методов связаны с UML-моделью, разработанной ранее.

На 8-й лабораторной работе

4. Для разработанной формы описать обработчики нажатия на все кнопки.

Для вариантов 1, 5: текстовый редактор:

– поле ввода текста должно поддерживать построчный ввод и реагировать на нажатие кнопки «Enter» созданием новой строки;

– все 12 кнопок по операциям редактирования должны быть снабжены обработчиками и обеспечивать, как минимум, следующие функции: копирование текста в буфер, вставка из буфера, сохранение в файл, загрузка из файла, просмотр буфера, поиск по слову;

– строки состояния с надписями должны выводить: число символов в тексте, число слов в тексте, число строк в тексте, число знаков препинания в тексте и обновляться по результатам любого изменения текста.

Для вариантов 2, 6: графический редактор:

– поле ввода графики должно поддерживать ввод графических примитивов (линия, треугольник, квадрат, круг, точка и т.д.) и поддерживать их редактируемость (поворот, растяжение, сжатие) до нажатия кнопки «Enter», должен быть режим рисования кривой мышью;

– все 14 кнопок по операциям редактирования должны быть снабжены обработчиками и обеспечивать, как минимум, следующие функции: ввод, как минимум, 6 видов примитивов, ввод как минимум следующих операций: «Сохранить», «Загрузить», «Очистить», «Выделить всё», «Снять выделение», «Залить цветом по номеру»;

– строки состояния с надписями должны выводить: число примитивов, число использованных цветов, время от начала редактирования (в минутах) и обновляться по результатам любого изменения графики.

Для вариантов 3, 7: анкета регистрации на сайте интересов:

– надпись с названием анкеты должна быть в прямоугольнике для 3-го варианта синего цвета с текстом шрифтом жёлтого цвета, для 7-го оранжевого цвета с текстом шрифтом чёрного цвета;

– все 10 полей ввода с 10 уточняющими надписями должны быть выровнены по левому краю и пущены в 2 ряда, для каждого поля должна быть проверка на корректность ввода и допустимость диапазона; 2 - 3 поля должны быть не активны до заполнения одного из других полей (например, пока не введено значение поля «Физическое лицо» не активно поле «ФИО»);

– для всех 4-х кнопок должны быть реализованы функции сохранения в файл анкеты, открытия дополнительных полей, очистки содержимого полей, выхода из программы без сохранения;

– строки состояния должны выводить: число заполненных полей и время начала ввода.

Для вариантов 4, 8: анкета заказа:

4) строка состояния с 3-мя надписями: «Корзина заполнена?», «Стоимость заказа», «Номер заказа» с выравниванием снизу.

– надпись с названием ресторана должна быть в овале для 4-го варианта зелёного цвета с текстом шрифтом тёмно-красного цвета, для 8-го коричневого цвета с текстом шрифтом белого цвета;

– все 8 полей ввода с 8 уточняющими надписями должны быть выровнены по правому краю и пущены в 1 ряд, для каждого поля должна быть проверка на корректность ввода и допустимость диапазона; 1 - 2 поля должны быть не активны до заполнения одного из других полей (например, пока не введено значение поля «Физическое лицо» не активно поле «ФИО»);

– для всех 5-ти кнопок должны быть реализованы функции сохранения в текстовый файл заказа, вызов окна с описанием выбранных блюд в виде списка надписей по числу выбранных блюд, очистки содержимого полей, сохранения и выхода, выхода из программы без сохранения;

– строки состояния должны выводить: состояние корзины (заполнена или нет), сумма заказа в рублях, номер заказа по порядку (счётчик должен сохраняться после выхода из программы).

Для вариантов 9, 10: форма ввода результатов эксперимента (по получению корней кубического уравнения $ax^3+by^2+cz+d=0$ методами подбора и по формуле Кардано):

– все 12 полей ввода с 15 уточняющими надписями должны быть выровнены по ширине и размещены наиболее компактно и удобно для восприятия; среди полей должно быть поле со списком числа итераций, поле со списком уровней моделирования, поле со списком даты и поле с шаблоном для времени, поле с заданным результатом моделирования, а также поле со строковым шаблоном для ФИО исследователей;

– все 7 кнопок меню должны быть снабжены обработчиками и обеспечивать, как минимум, следующие функции: запуск моделирования, останов моделирования (пауза), задание числа итераций (разблокировка поля), задание уровня (разблокировка поля), задание формулы (вызов формы для формирования расчётов), выбор метода моделирования (повторение в цикле, проход один раз, проход до достижения нужного результата), выход;

– строки состояния должны выводить: время моделирования в секундах в виде счётчика, изменяемого при моделировании, число рассчитанных значений, «Число итераций», «Номер уровня» с выравниванием снизу.

Вариант доп. операций получить у преподавателя.

В заголовке формы должны быть указаны Фамилии и инициалы участников группы.

На форме должны быть перечислены все особенности варианта с указанием того, какие настройки по варианту для каждого задания. Например,

«Задание 1 – Иванов И.И., Петров П.П.

Задание 2 – Автомобили, 5-й вариант, 01.01.2101

Задание 3 – цвет № 100, расстояние 25 пт, тип сообщения PLAIN_MESSAGE

…

Задание 7 – факториал суммы ASCII-кодов первых букв фамилий участников группы + 77, тип сообщения QUESTION_MESSAGE.

5. Одна из кнопок должна по нажатию менять цвет всех кнопок на цвет с номером 20*<номер варианта> и увеличивать расстояние между панелями на 5*<номер варианта> пикселей с выдачей сообщения о выполнении. Тип сообщений по вариантам:

1, 6 варианты – ERROR_MESSAGE;

2, 7 варианты – INFROMATION_MESSAGE;

3, 9 варианты – WARNING_MESSAGE;

4, 8 варианты – QUESTION_MESSAGE;

5, 10 варианты – PLAIN_MESSAGE.

У чётных вариантов все кнопки должны выравниваться по верхнему краю, а у нечётных по нижнему краю панели.

6. Одна из кнопок должна обеспечивать выход из программы. Перед выходом должно выдаться сообщение с результатом выполнения (до закрытия сообщения программа не должна закрываться, основная форма должна находиться в неактивном состоянии позади сообщения):

1, 6 варианты: сумму третьих степеней ASCII-кодов первых букв имён участников группы – 14;

2, 7 варианты: корень третьей степени из суммы ASCII-кодов первых букв фамилий участников группы + 121;

3, 8 варианты: разность большего и меньшего от произведений ASCII-кодов первой и последних букв имён двоих участников группы +33;

4, 9 варианты: сумму десятичных логарифмов ASCII-кодов вторых букв фамилий участников группы – 27;

5, 10 варианты: факториал суммы ASCII-кодов первых букв фамилий участников группы + 77.

Расчёт должен быть реализован формулой. Код ASCII должен быть взят функцией (например, getBytes).

Тип сообщений по вариантам:

1, 3 варианты – INFROMATION_MESSAGE;

2, 4 варианты – ERROR_MESSAGE;

5, 6 варианты – QUESTION_MESSAGE;

7, 8 варианты – WARNING_MESSAGE;

9, 10 варианты – PLAIN_MESSAGE.

4 Контрольные вопросы

1 Какие обработчики событий Вы знаете?

2 Что такое источник события?

3 Что такое объект listener?