Разработка приложения Калькулятор

Создадим простое приложение – калькулятор на базе Swing.

Чтобы начать создание первого Java-проекта, выполним команду **File, New, Project...** , рис.2.

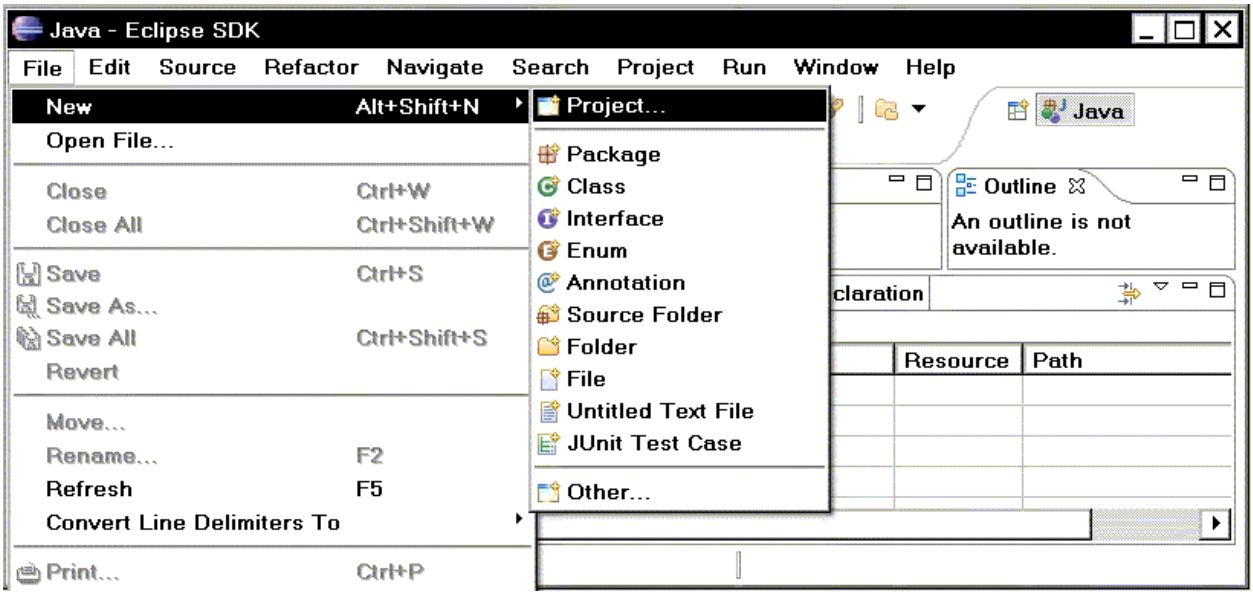

Рис.1. Создание нового проекта

Откроется окно со списком типов проектов, рис. 2, выберем **Java Project**. Затем нажимем кнопку **Next** и переходим во второе окно мастера, рис.3.

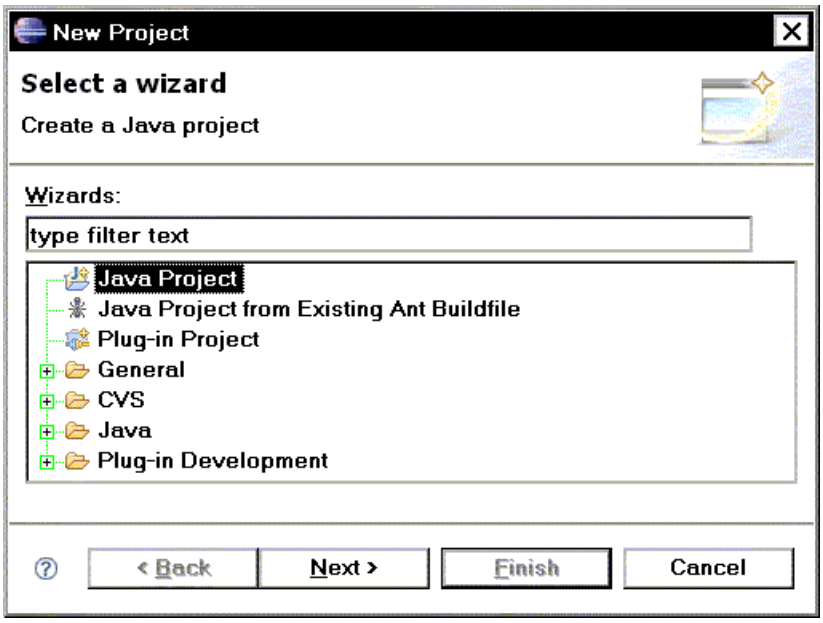

Рис.2. Выбор типа создаваемого проекта

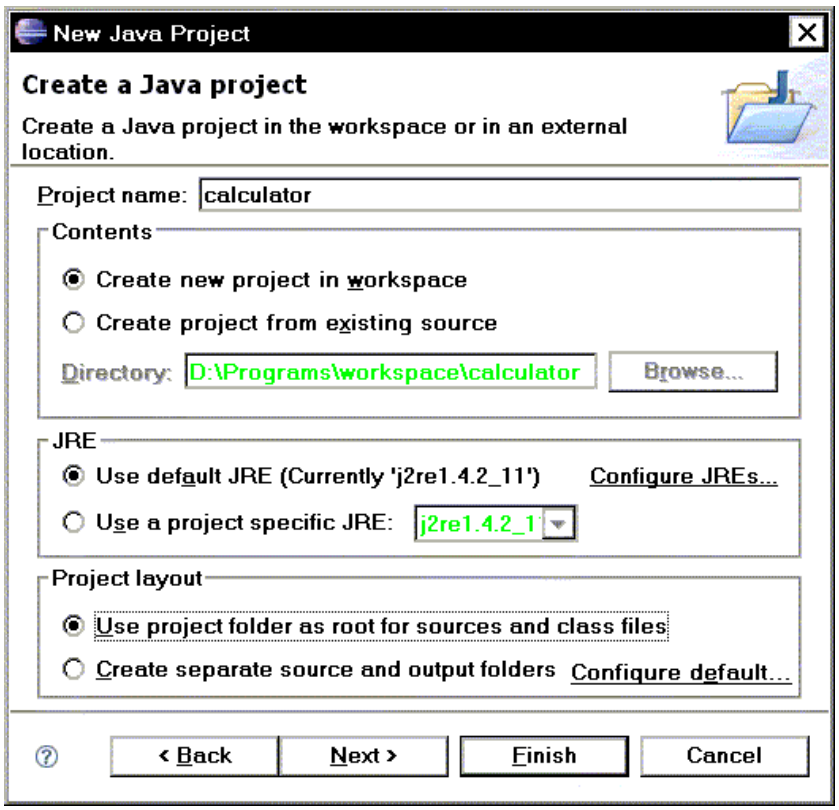

Рис.3. Второй диалог мастера проектов. Ввод имени проекта

2. Во втором диалоге, рис.3, в качестве названия проекта введите *Calculator* и снова нажмите **Next**. Здесь же можно указать место для хранения исходных и скомпилированных файлов, а также создать отдельные каталоги для таких файлов.

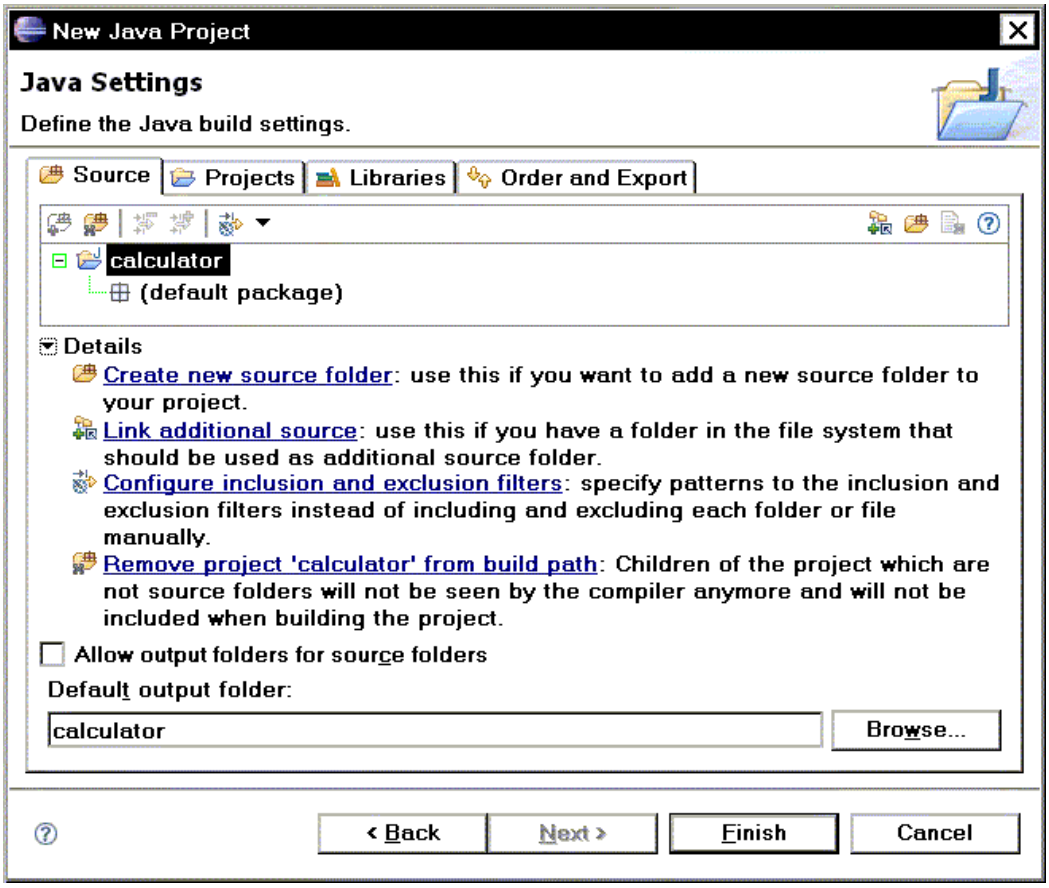

Рис.4. Установки нового проекта

3. Последний шаг мастера, рис. 4, позволяет указать место для хранения исходных и скомпилированных файлов, а также задать любые подпроекты и библиотеки, которые могут понадобиться для работы и компоновки текущего проекта. Нажмем **Create new source folder**, откроется окно в котором введем имя папки для исходного кода, рис. 5. Нажмем, затем кнопку Finish, после чего вернемся в окно Java Setting, рис. 4, которое примет вид, показанный на рис.6.

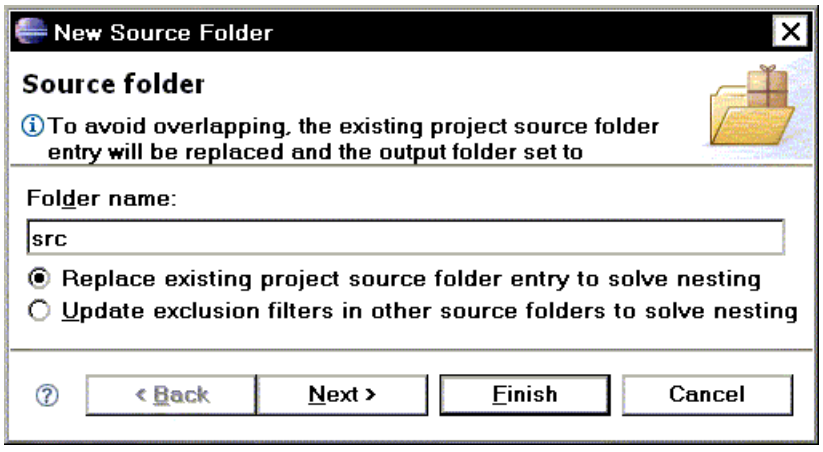

Рис.5. Ввод имени папки для исходного кода

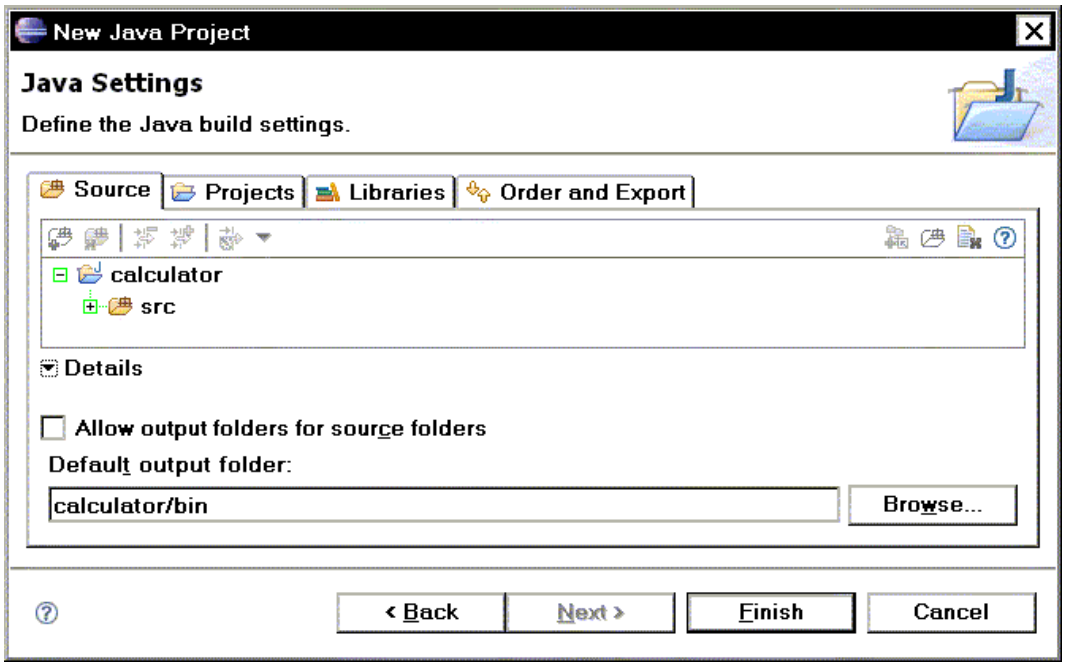

Рис.6. Завершение создания проекта

4. Щелкните на кнопке **Finish.** В появившемся окошке, рис.7, предлагающем переключить перспективу жмем на YES и Eclipse создаст новый проект.

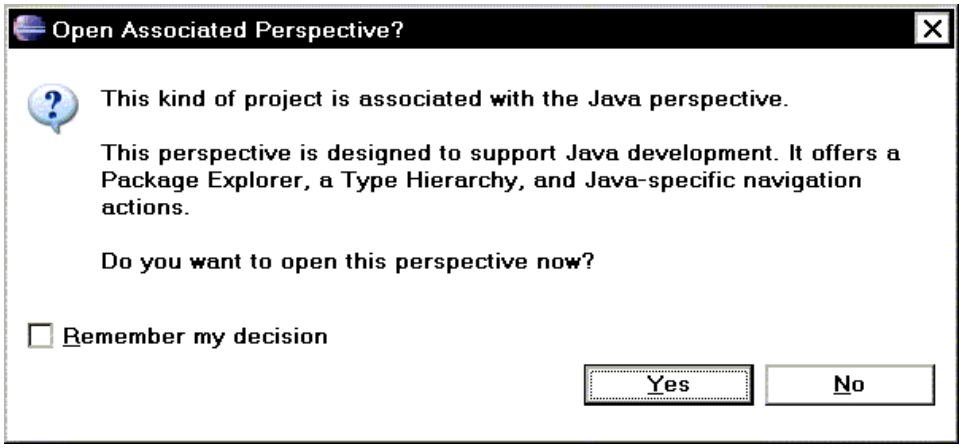

Рис.7. Предложение переключиться на соответствующую перспективу

После создания проекта, вы можете обратить внимание на то, что окно Eclipse выглядит не совсем так, как вначале, рис.8.

Новый внешний вид носит название **Перспективы Java**. Перспектива, в терминах Eclipse, это сохраняемый внешний вид окна, включающий любое число редакторов (editors) и представлений (views). В поставку Eclipse входит несколько перспектив по умолчанию (Java, Debug, Resource и так далее), которые можно настраивать. Вы также можете создавать новые перспективы. Перспективы управляются командами меню Window.

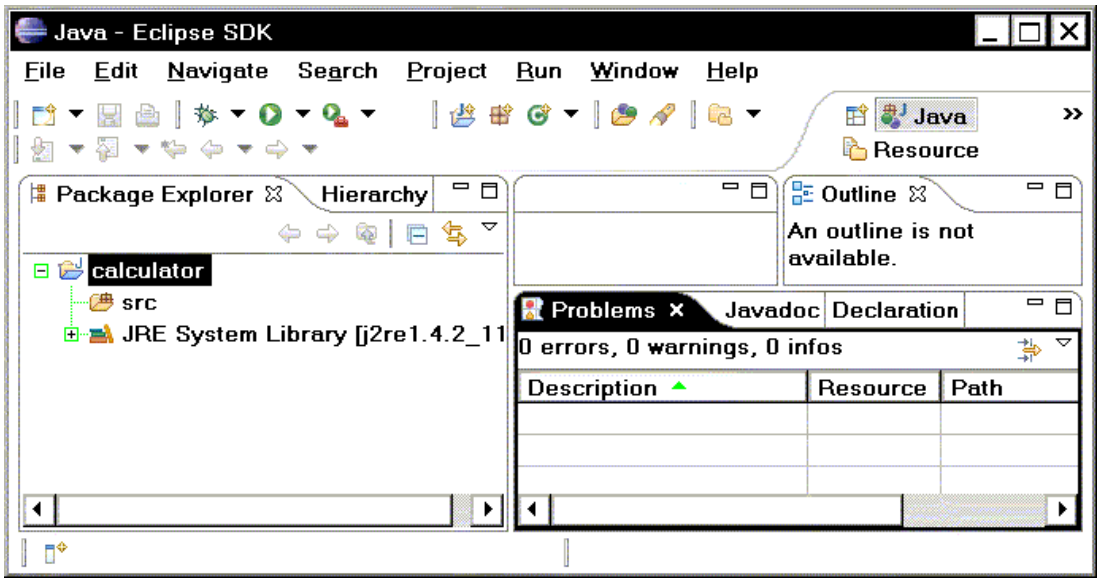

Рис.8. Перспектива Java

Следующим шагом в создании нашего Java проекта станет создание папок, в которых будет содержаться исходный код. Выполним команду **Window, Open Perspective, Other** и выберем в окне Open Perspective, рис.9, перспективу ресурсов (**Resource**).

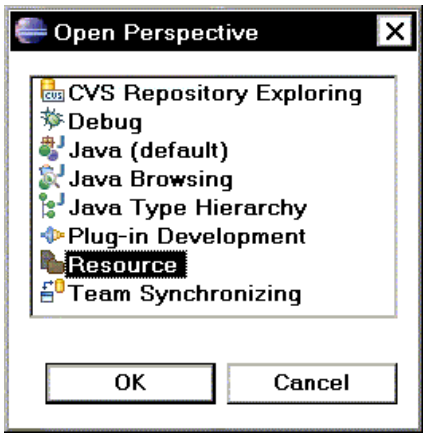

Рис.9. Выбор перспективы

В представлении Навигатора Файлов, рис.10 раскройте дерево папок и найдите узел *src*. Выберите этот узел, а затем пункт меню **File, New, Folder**. В открывшемся диалоге убедитесь, что выбрана папка *src*, затем введите *com* в поле **Folder Name**.

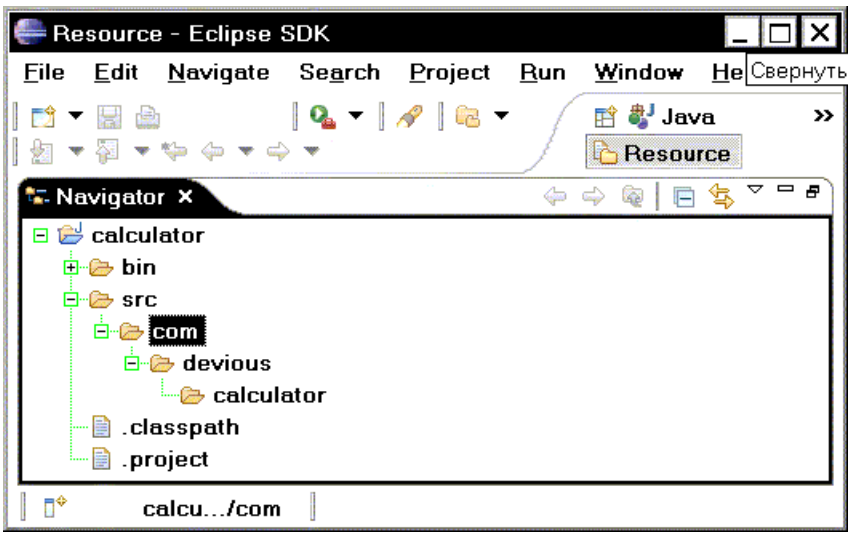

Рис.10. Представление Навигатора

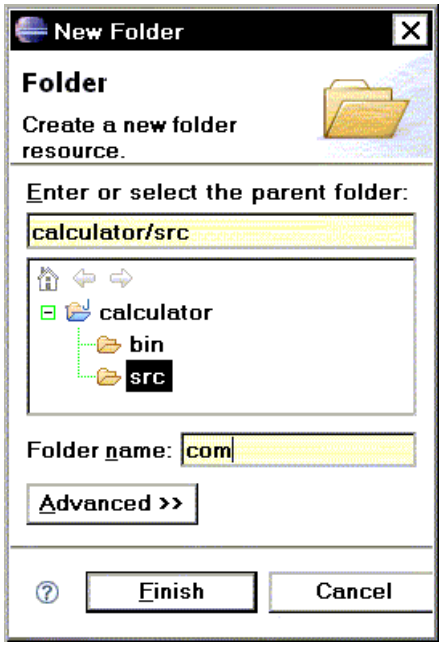

Рис.11. Создание папки в проекте

Создание большого числа папок с использованием диалога New Folder может показаться несколько неудобным. К счастью, Eclipse позволяет сделать эту работу, используя методы, принятые в вашей операционной системе: командную оболочку, Windows Explorer и так далее. Воспользуйтесь одним из этих методов и создайте подпапки *devious* и *calculator* в папке *com*. На рис.12 показано окно Проводника Windows, с созданными в нем папками.

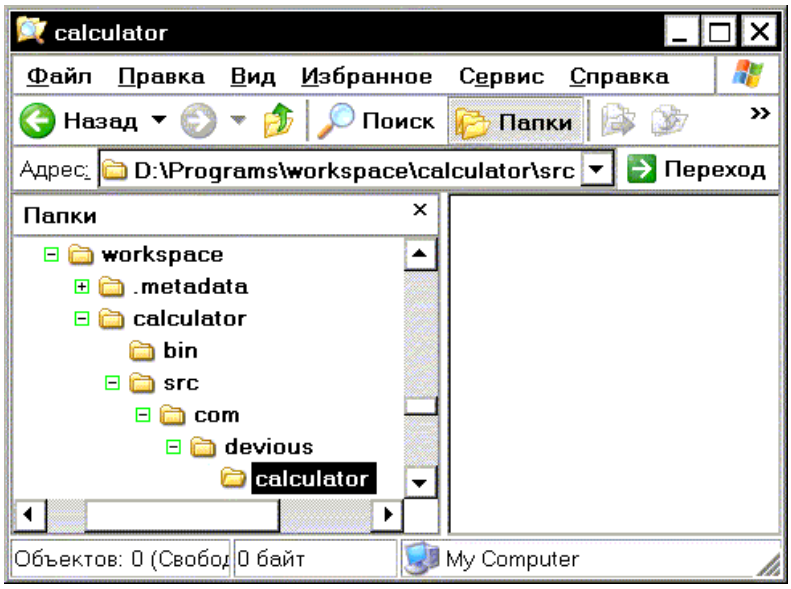

Рис.12. Создание папок проекта в Проводнике Windows

После этого выберите любую папку в представлении Навигатора, рис.10, а затем пункт меню **File, Refresh**. Eclipse просмотрит файловую систему и приведет проект в соответствие с внесенными изменениями. На данный момент наш проект должен выглядеть примерно так, как представлено на рис.10.

И последнее замечание по созданию файлов: на ранних этапах разработки проекта вы можете столкнуться с необходимостями часто запускать код на выполнение, создавать новые файлы или папки, добавлять новый код, компилировать и тестировать его снова. Важно помнить, что меню Eclipse отображается по-разному в зависимости от текущей перспективы. Таким образом, если вы переключаетесь в перспективу ресурсов для создания новых файлов или папок через представление Навигатора, вы обнаружите, что например, меню **Run** заметно отличается от аналогичного меню в перспективе Java. Одним из решений данной проблемы может послужить использование пункта меню **Window, Show View** и отображение представления Навигатора в перспективе Java. Чтобы закрепить это изменение, используйте команду меню **Window, Save Perspective As...**, позволяющее сохранить настройки перспективы под новым именем или заменить настройки существующей перспективы Java.

1.1. Добавление кода

Теперь, когда мы построили структуру папок, мы можем начать добавлять код к нашему проекту. Мы разделим нашу программу калькулятора на три отдельных файла кода: *CalcMode.java*, *CalcPanel.java* и *Calculator.java*.

Выберем в представлении Навигатора папку *calculator* и выполним команду **File, New, File** и в окне **New File**, рис.13, введем имя создаваемого файла *CalcModel.java*.

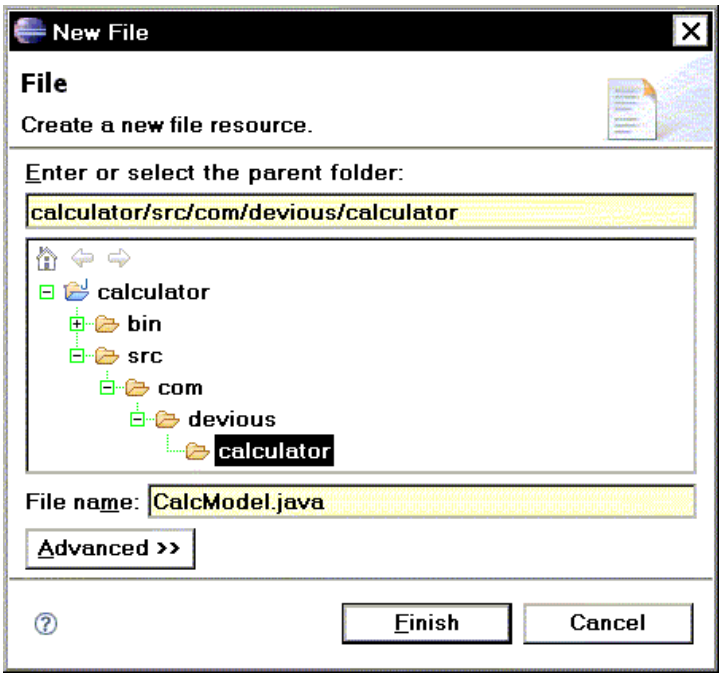

Рис.13. Создание нового файла

После нажатия кнопки **Finish** возвращаемся в перспективу, где появится окно редактора кода с открытым файлом *CalcModel.java*, рис. 14.

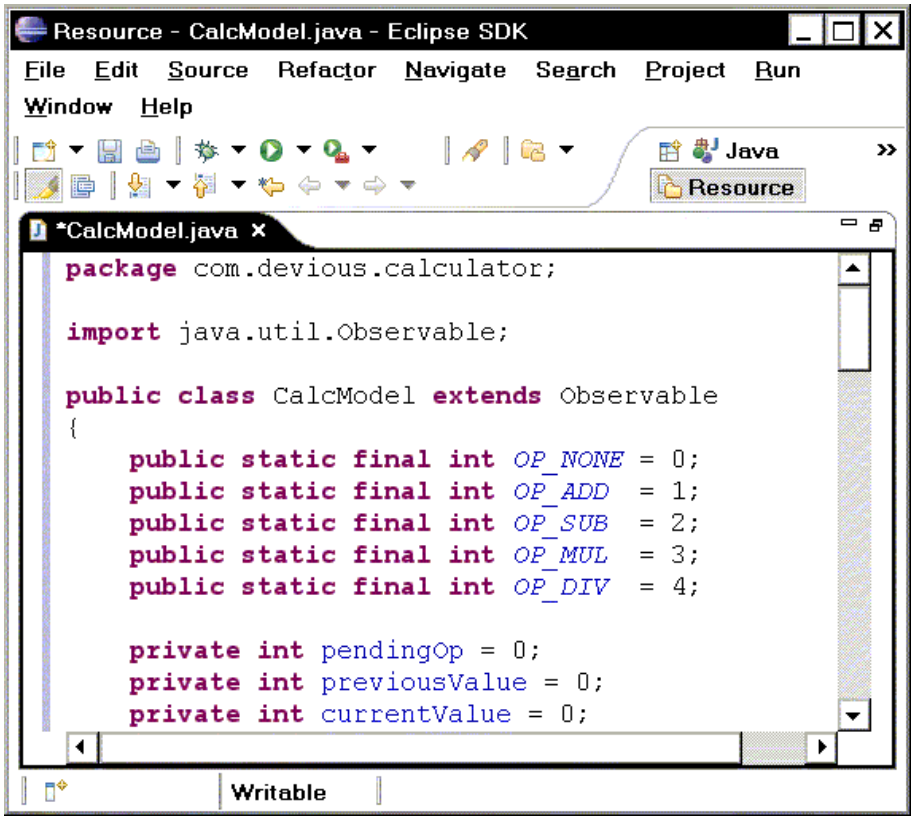

Рис.14. Окно редактора кода программы

Введем в окно кода следующий программный код:

CalcModel.java

```
package com.devious.calculator;
```

```
import java.util.Observable;

public class CalcModel extends Observable
{
    public static final int OP_NONE = 0;
    public static final int OP_ADD  = 1;
    public static final int OP_SUB  = 2;
    public static final int OP_MUL  = 3;
    public static final int OP_DIV  = 4;

    private int pendingOp = 0;
    private int previousValue = 0;
    private int currentValue = 0;

    public String getValue()
    {
        return Integer.toString(currentValue);
    }
    public void addDigit(String s)
    {
        char c = s.charAt(0);
        String val = getValue() + c;
        setDisplay(Integer.parseInt(val));
    }
    public void addDigit(char c)
    {
      String val = getValue() + c;
      setDisplay(Integer.parseInt(val));
    }
    public void setOperation(int op)
    {
        if (pendingOp != OP_NONE)
            calculate();
        previousValue = currentValue;
        this.pendingOp = op;
        currentValue = 0;
    }
    public void setDisplay(int value)
    {
        currentValue = value;
        setChanged();
        notifyObservers();
    }
    public void clear()
    {
        this.pendingOp = OP_NONE;
        previousValue = 0;
        setDisplay(0);
    }
    public void calculate()
    {
        switch (pendingOp)
        {
        case OP_ADD:
            setDisplay(previousValue + currentValue);
            break;
        case OP_SUB:
            setDisplay(previousValue - currentValue);
            break;
        case OP_MUL:
```

```
            setDisplay(previousValue * currentValue);
            break;
        case OP_DIV:
            setDisplay(previousValue / currentValue);
            break;
        }
        pendingOp = OP_NONE;
        previousValue = 0;
    }
}
```

Аналогично создадим файл:

CalcPanel.java

```
package com.devious.calculator;

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

public class CalcPanel
      extends JPanel
      implements ActionListener, Observer
{
    private CalcModel model;
    private JLabel display = new JLabel("0");
    public CalcPanel(CalcModel model)
    {
        super();
        this.model = model;
        model.addObserver(this);
    }
    protected JButton addButton(String label)
    {
        JButton b = new JButton(label);
        b.addActionListener(this);
        return b;
    }
    protected JPanel createButtonPanel()
    {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(0, 4));

        panel.add(addButton("1"));
        panel.add(addButton("2"));
        panel.add(addButton("3"));
        panel.add(addButton("+"));

        panel.add(addButton("4"));
        panel.add(addButton("5"));
        panel.add(addButton("6"));
        panel.add(addButton("-"));

        panel.add(addButton("7"));
        panel.add(addButton("8"));
        panel.add(addButton("9"));
        panel.add(addButton("*"));

        panel.add(addButton("C"));
```

```
        panel.add(addButton("0"));
        panel.add(addButton("="));
        panel.add(addButton("/"));

        return panel;
    }
    public void init()
    {
        this.setLayout(new BorderLayout());
        display.setVerticalTextPosition(JLabel.CENTER);
        display.setHorizontalTextPosition(JLabel.RIGHT);
        display.setVerticalAlignment(JLabel.TOP);
        display.setHorizontalAlignment(JLabel.TRAILING);
        display.setBorder(BorderFactory.createLineBorder(Color.black));

        this.add(display, BorderLayout.NORTH);
        this.add(createButtonPanel(), BorderLayout.CENTER);
    }
    public void actionPerformed(ActionEvent evt)
    {
        System.out.println(evt.paramString());
        try {
            switch (evt.getActionCommand().charAt(0)) {
            case '+':
                model.setOperation(CalcModel.OP_ADD);
                break;
            case '-':
                model.setOperation(CalcModel.OP_SUB);
                break;
            case '*':
                model.setOperation(CalcModel.OP_MUL);
                break;
            case '/':
                model.setOperation(CalcModel.OP_DIV);
                break;
            case '=':
                model.calculate();
                break;
            case 'C':
                model.clear();
                break;
            default:
                model.addDigit(evt.getActionCommand());
                break;
            }
        } catch (NumberFormatException ex) {}
    }
    public void update(Observable o, Object arg)
    {
        display.setText(model.getValue());
    }

}
```

Далее создаем файл

Calculator.java

```
package com.devious.calculator;

import javax.swing.*;
```

```
public class Calculator
{
    public static void main(String[] args)
    {
        try {
            UIManager.setLookAndFeel(
                UIManager.getCrossPlatformLookAndFeelClassName());
        } catch (Exception e) {}

        JFrame calcFrame = new JFrame("Calculator");

        CalcModel model = new CalcModel();
        CalcPanel calcPanel = new CalcPanel(model);
        calcPanel.init();

        calcFrame.setContentPane(calcPanel);
        calcFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        calcFrame.pack();
        calcFrame.setVisible(true);
    }
}
```

После создания файлов перейдем в представление Навигатора и выполним команду команду **File, Refresh** для обновления проекта Eclipse, рис.15.

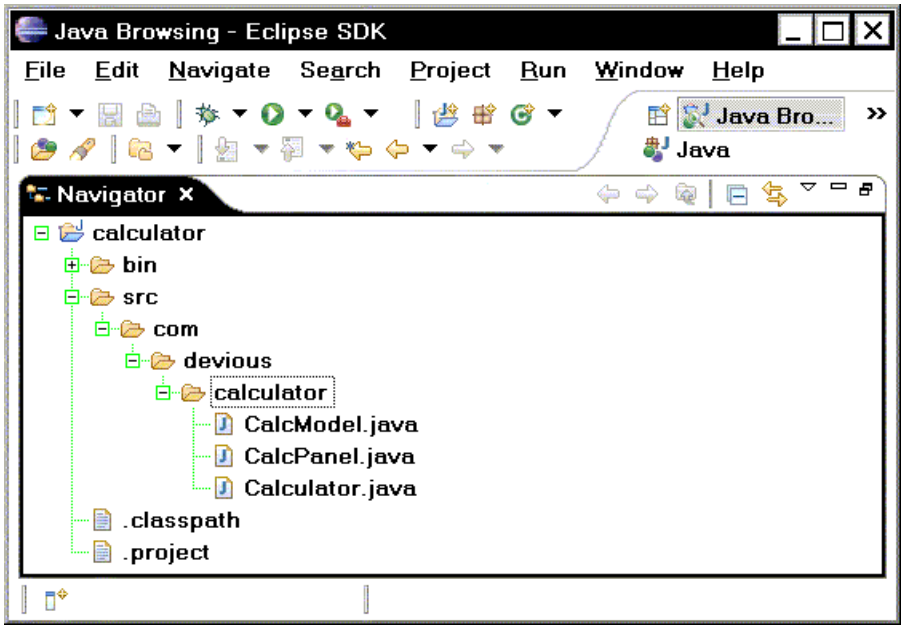

Рис.15. Файлы в составе проекта

CalcPanel.java обрабатывает большую часть пользовательского интерфейса нашего калькулятора. *Calculator.java* отвечает за настойку графического интерфейса и его отображение. Большая часть важной работы обеспечивается файлом *CalcModel.java*, который реализует непосредственно "механику" калькулятора: реакцию на события, производство числовых вычислений, обновление окна и так далее.

Самый простой путь исследования кода заключается в переключении в перспективу Java или в перспективу Просмотра Java (Java Browsing). Используйте элемент меню

Window, Open Perspective или нажмите кнопку **Open** на "перспективной" панели инструментов.

При использовании **Package Explorer** для просмотра исходного кода вы можете заметить, что представление группировки кода является избыточным. Чтобы его закрыть, нажмите на значок x в заголовке его окна. При необходимости отобразить его снова, используйте команду меню **Window, Show View, Outline**. На рис.16 показано содержимое пункта меню **Window, Show View** и окна кода.

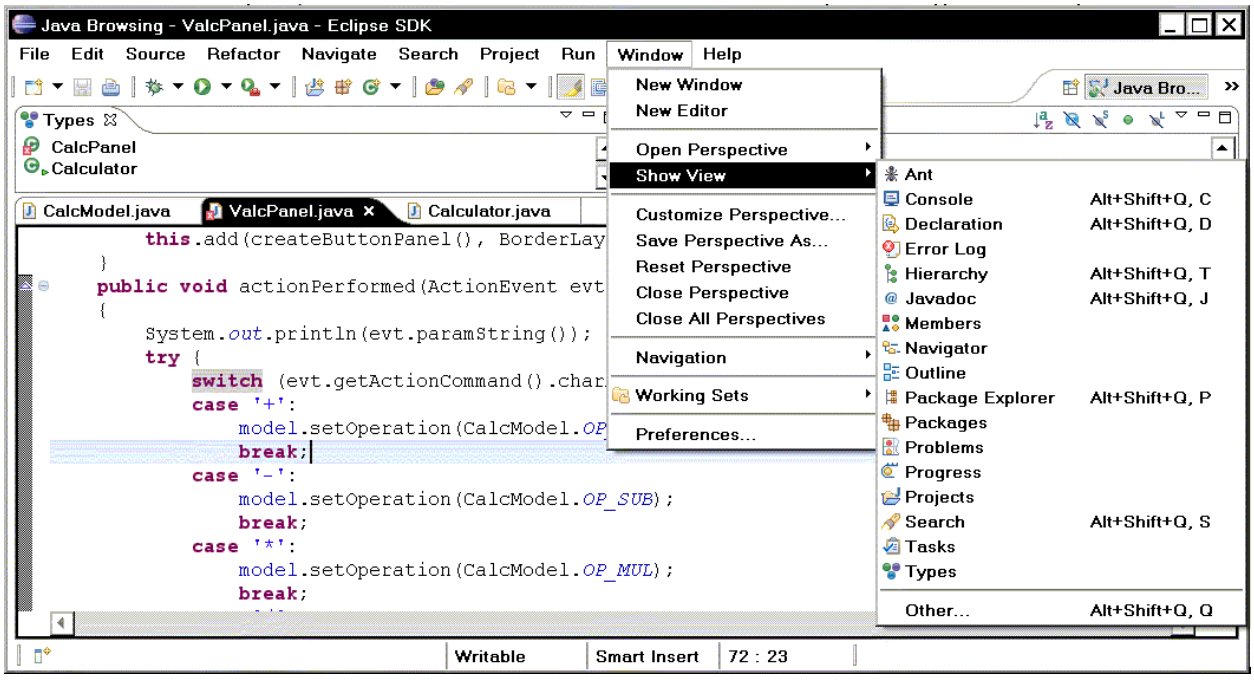

Рис.16. Выбор представления для исследования кода

1.2. Запуск кода и отладка

Итак установили Eclipse, создали Java-проект, добавили папки и файлы, необходимые для работы простого приложения. Самое время посмотреть, как оно будет работать. Для выполнения последующих инструкций по запуску приложения вам нужно будет перейти в перспективу Java:

1. Из меню **Run** выберите элемент **Run...**

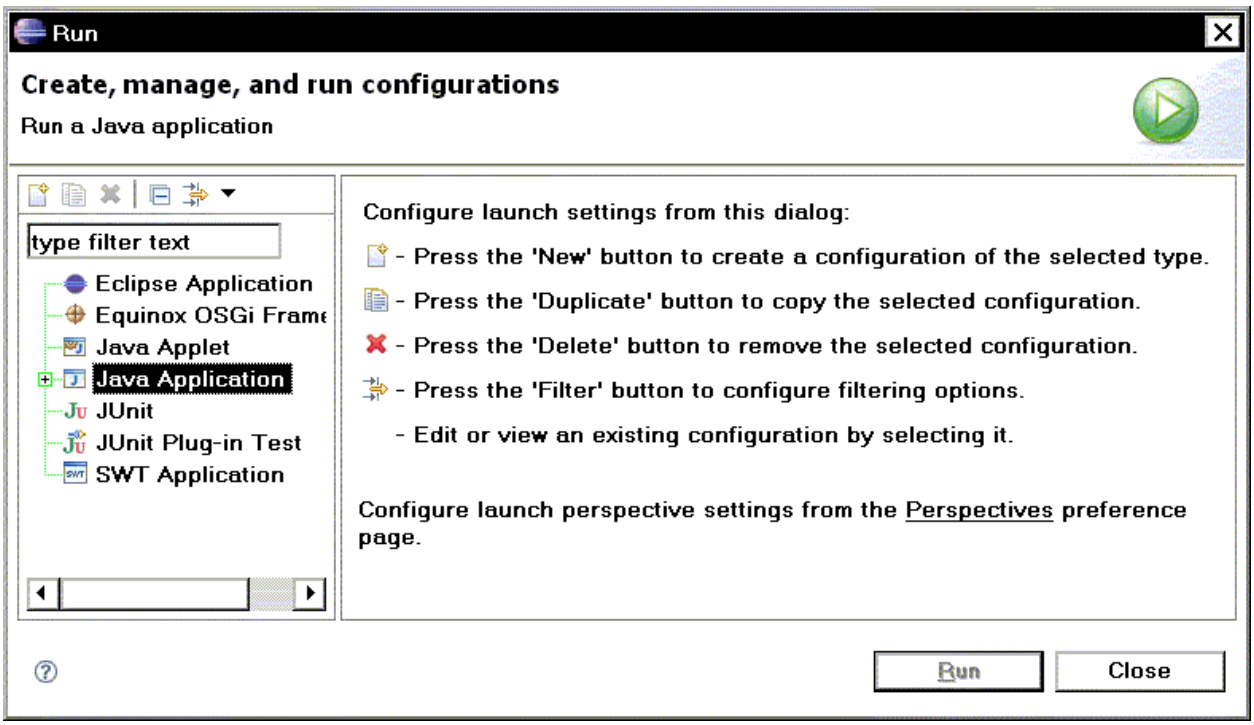

Рис.17. Выбор конфигурации проекта

2. В появившемся диалоге, рис.17, из списка Конфигураций Запуска (**Launch Configurations**) выберите "Java Application", затем нажмите на кнопку **New**. Откроется окно, рис.18, в которм надо настроить проект.

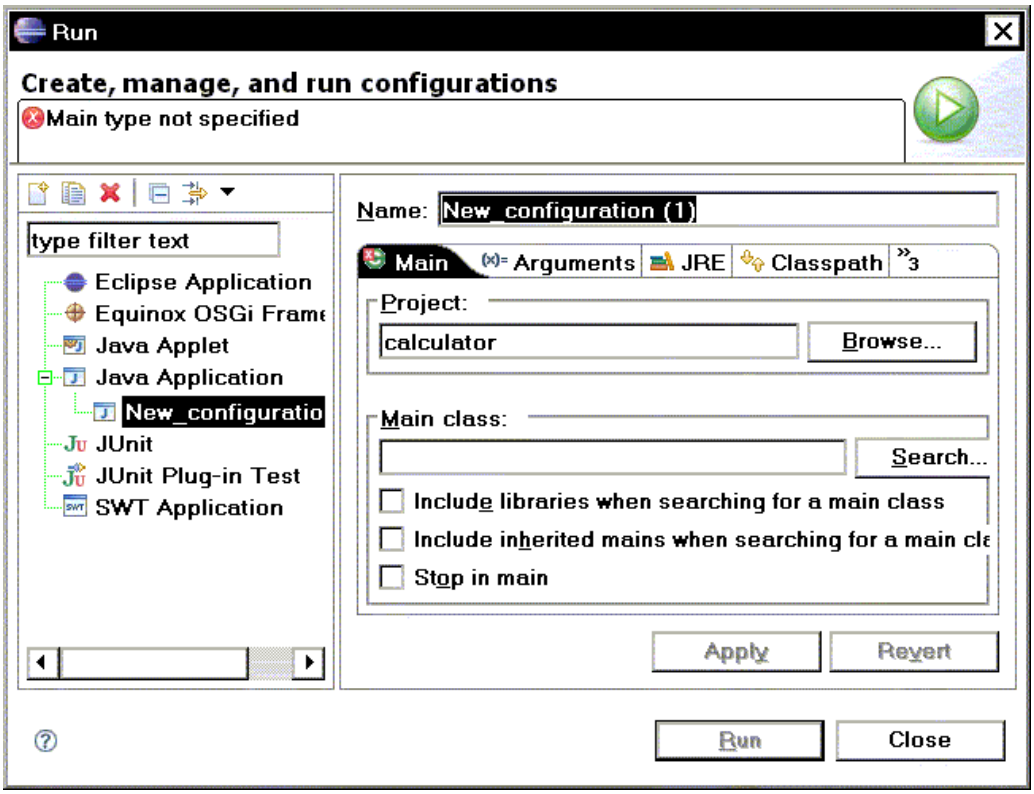

Рис.18. Окно для настройки конфигурации проекта

3. В поле **Name**, рис.18, вместо *New_configuration (1)* введем *Calculator*.

4. Нажмите кнопку **Search...**. Откроется окно **Select Main Type**, рис.19, и выберите *Calculator* в качестве главного типа (**Main types**).

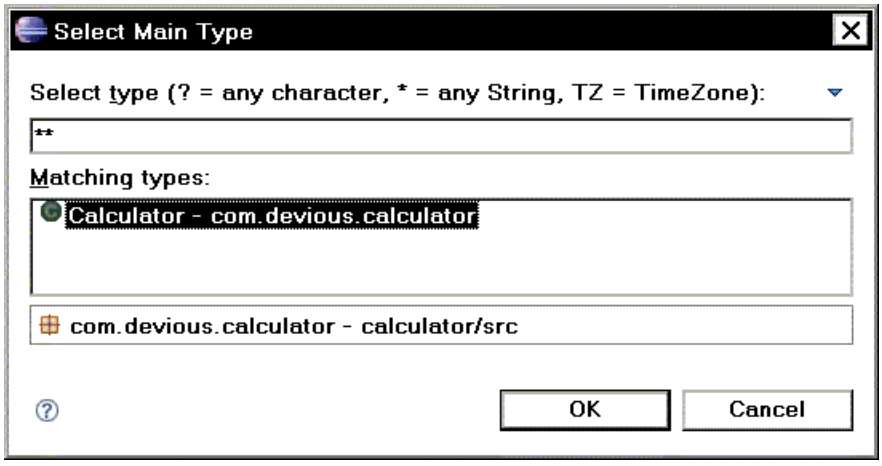

Рис.19. Выбор главного класса

Нажмем в окне **Select Main Type** кнопку OK, вернемся в окно **Run**, рис.20. В поле Main Class будет: *com.delirious.calculator.Calculator*

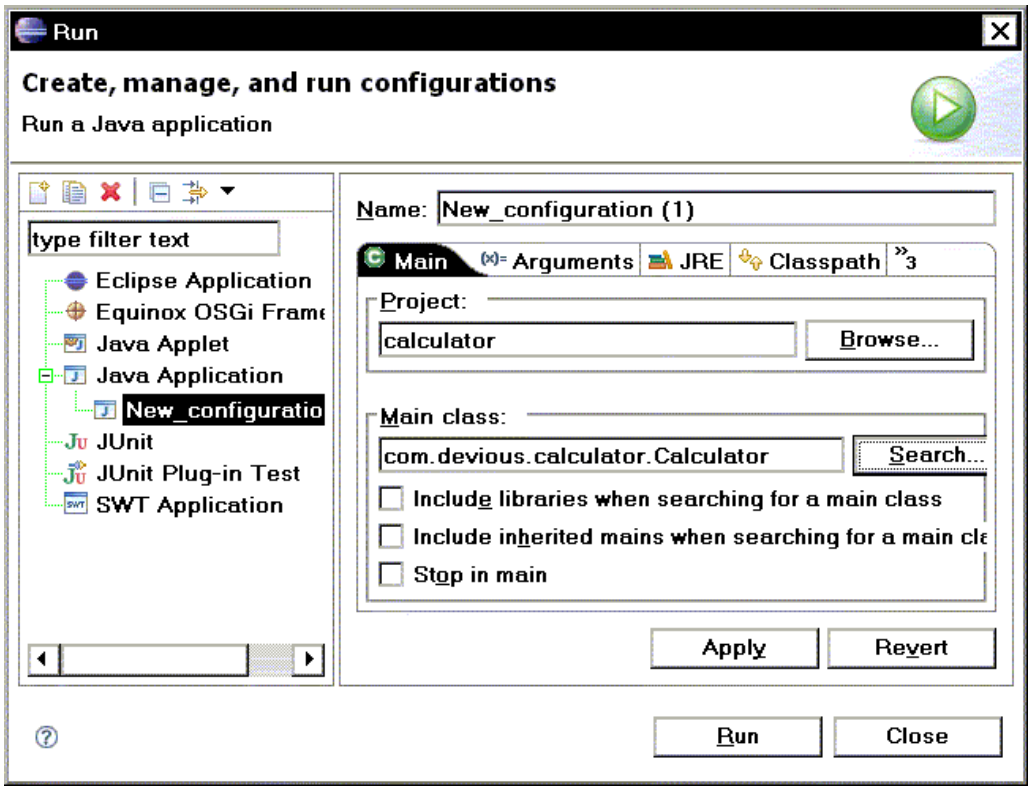

Рис.20. Настроенная конфигурация проекта

5. Щелкните кнопку **Run** для сохранения конфигурации и запуска приложения. Программа выполнится, на экране появится кальеулятор, рис.21.

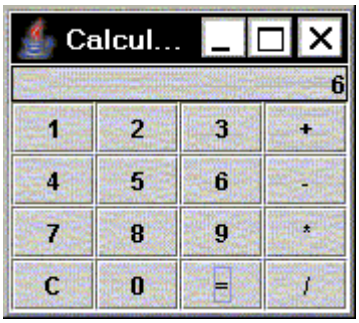

Рис.21. Работающая программа