Оглавление

[**Введение** 2](#_Toc106811892)

[**1 Разработка компьютерной игры** 3](#_Toc106811893)

[**1.1 Разработка технического задания** 3](#_Toc106811894)

[*1.1.1 Полное наименование системы и её условное обозначение* 3](#_Toc106811895)

[*1.1.2 Основание для разработки* 3](#_Toc106811896)

[*1.1.3 Назначение и цели создания системы* 3](#_Toc106811897)

[**1.2 Анализ технического задания** 4](#_Toc106811898)

[*1.2.1 Общие положения* 4](#_Toc106811899)

[*1.2.2 Требования к системе* 4](#_Toc106811900)

[**1.3 Стандарты** 5](#_Toc106811901)

[**1.4 Требования к техническим средствам** 5](#_Toc106811902)

[**2 Разработка модели предметной области** 6](#_Toc106811903)

[**2.1 Анализ предметной области** 6](#_Toc106811904)

[**2.2 Разработка структуры классов** 6](#_Toc106811905)

[**3 Разработка структуры приложения** 7](#_Toc106811906)

[**3.1 Разработка архитектуры** 7](#_Toc106811907)

[**3.2 Проектирование пользовательского интерфейса и взаимодействие с ним** 7](#_Toc106811908)

[**Заключение** 10](#_Toc106811909)

[**Список использованной литературы** 1](#_Toc106811909)1

[**Приложение** 1](#_Toc106811911)2

**Введение**

В данной пояснительной записке рассматривается описание приложения «Калькулятор» на основе объектно-ориентированного подхода.

При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

В отличие от других подходов к программированию, объектно-ориентированный подход требует глубокого понимания основных принципов или концепций, на которых он базируется.

К числу основополагающих понятий ООП обычно относят абстракцию данных, наследование, инкапсуляцию и полиморфизм.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования.

В качестве основного инструмента разработки применяется NetBeans IDE 12.6. Язык программирования Java.

**1 Разработка компьютерной игры**

**1.1 Разработка технического задания**

*1.1.1 Полное наименование системы и её условное обозначение*

Полное наименование: Разработка приложения «Калькулятор» для помощи в изучении принципов решения несложных математических задач учениками начальной школы.

Условное обозначение системы: «Калькулятор».

*1.1.2 Основание для разработки*

Основанием для разработки данной информационной системы является приказ по НГТУ на выпускную квалификационную работу по дисциплине «Разработчик цифровых продуктов».

*1.1.3 Назначение и цели создания системы*

1.1.3.1 Назначение системы

Программный продукт предназначен для помощи в изучении принципов решения несложных математических задач (сложение, вычитание, произведение, деление) учениками начальной школы.

1.1.3.2 Цели создания системы

Программный продукт разрабатывается с целью:

1) изучения работы калькулятора на базовом уровне;

2) мотивации в решении простых математических задач с целью получения верного ответа среди учеников начальной школы;

3) облегчения работы учителя путём уменьшения числа поступающих от учеников вопросов о проверке ответов.

С помощью приложения пользователь сможет достигнуть вышеперечисленные цели.

1.1.3.3 Основные задачи разработки

1) Обеспечить вход пользователя в приложение;

2) Обеспечить запись и удаление задач;

3) Обеспечить хранение данных приложения во время его работы;

4) Обеспечить возможность вычислять несложные арифметические задачи (сложение, вычитание, произведение, деление).

**1.2 Анализ технического задания**

*1.2.1 Общие положения*

Согласно техническому заданию необходимо разработать приложение для компьютера, которое предназначено для решения несложных математических задач, с использованием фонового изображения в «успокаивающих» тонах с целью сохранения нейтрального психологического состояния.

*1.2.2 Требования к системе*

1.2.2.1 Требования к структуре и функционированию системы

Программный продукт, разрабатываемый в рамках выпускной квалификационной работы, должен удовлетворять следующему перечню функциональных требований:

1. Удобный и понятный графический интерфейс;
2. Сохранение состояния приложения во время его работы;
3. Возможность ввести и удалить данные.

1.2.2.2 Входные данные

Входными данными при работе с программным продуктом должны быть нажатия на кнопки мыши (левая).

1.2.2.3 Выходные данные

Выходными данными при работе программы являются данные, выводимые на экран пользователя.

1.2.2.4 Требования к надёжности

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при возникновении внештатных ситуаций.

1.2.2.5 Требования к эргономике и технической эстетике

Подсистемы ввода данных, а также формирования и визуализации отчетности должны обеспечивать удобный для конечного пользователя интерфейс.

Главное окно программного продукта должно позволять пользователю увидеть текущие расчёты и их конечный результат.

**1.3 Стандарты**

Программный продукт разрабатывается на основании следующих государственных стандартов:

1. 2.103-68 ЕСКД. Стадии разработок
2. 2.104-68 ЕСКД. Основные надписи
3. 2.105-95 ЕСКД. Общие требования к текстовым документам
4. 2.111- 68 ЕСКД. Общие требования к текстовым документам.
5. 2.118-73 ЕСКД. Техническое предложение
6. 2.120-73 ЕСКД. Технический проект
7. 2.316-68 ЕСКД. Правила нанесения на чертежах надписей, технических требований и таблиц
8. 7.1-2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

**1.4 Требования к техническим средствам**

Для удобства работы система должна обеспечивать отображение *GUI* с расширенным набором пользовательских элементов, что соответствует платформе *NetBeans, Java* не ниже версии 8.0.2.

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

1. компьютер/процессор: компьютер-терминал с процессором класса *Intel* *Core i7* с тактовой частотой 2,3 ГГц и новее;
2. память: не менее 512 МБ ОЗУ;
3. монитор: монитор с разрешением 1920x1080 точек или более высоким;
4. наличие свободного дискового пространства на жёстком диске не менее 500 Мб.

**2 Разработка модели предметной области**

**2.1 Анализ предметной области**

Программируемое приложение должно позволять пользователю вводить и удалять данные, а также производить решение несложных арифметических задач и показывать их результат.

Нефункциональные требования к программному средству:

1. эффективность: программа должна иметь минимальные требования к аппаратному обеспечению. Не должно требоваться дополнительных периферийных средств (сканер, дополнительные дисководы и т. д.);
2. понятность: программа должна быть интуитивно понятна, иметь удобный пользовательский интерфейс;
3. эстетичность: программа не должна включать в себя лишние и ненужные данные, а также иметь нехарактерный внешний вид интерфейса.

**2.2 Разработка структуры классов**

В соответствие с анализом предметной области можно выделить основные классы информационной системы и их атрибуты:

1. Класс *NewJFrame* – класс, представляющий собой простой калькулятор, который можно использовать для выполнения базовых математических операций, таких как сложение, вычитание, произведение и деление;
2. Класс *Calc* вводит основную функцию *main* и не нуждается в последующем описании.

**3 Разработка структуры приложения**

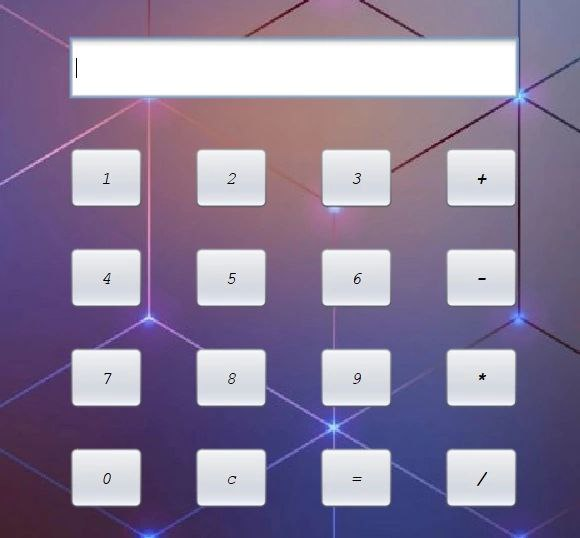
**3.1 Разработка архитектуры**

Так как «Калькулятор» - программа несложная, понадобится лишь один класс с описанием различных форм и алгоритма при совершении действия пользователем.

Класс *NewJFrame* содержит элементы пользовательского интерфейса Swing, такие как кнопки, текстовое поле и метку, а также методы для обработки событий, которые возникают при взаимодействии пользователя с калькулятором. Класс также содержит перечисление *Deist*, которое используется для определения текущей выбранной операции калькулятора.

**3.2 Проектирование пользовательского интерфейса и взаимодействие с ним**

В данном проекте для продуктивной работы реализован удобный пользовательский интерфейс (рис. 1). Фон интерфейса подобран в нейтральных цветах с использованием геометрических фигур.



**Рис. 1 – Интерфейс приложения**

Основной сценарий работы пользователя с приложением:

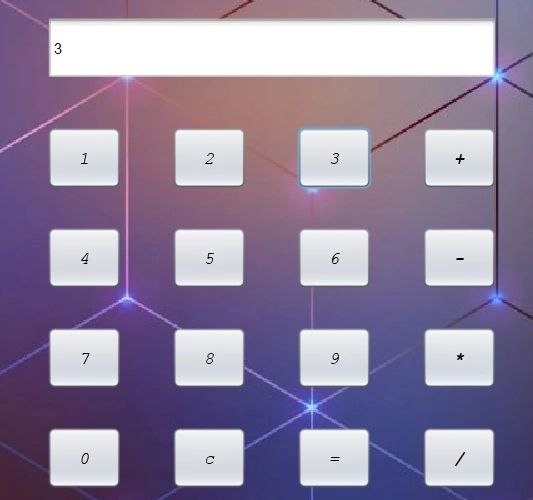
1. Пользователь видит окно «Калькулятора» с кнопками ввода данных – цифрами и знаками функций;
2. Левой кнопкой мыши пользователь нажимает на кнопку с нужным ему значением (цифра, знак функции), повторяя это действие до достижения нужной формулы;
3. При необходимости удалить введённое значение пользователь нажимает на соответствующую кнопку панели приложения;
4. При необходимости получить результат пользователь нажимает на соответствующую кнопку панели приложения.

Как пример, рассмотрим произведение чисел 2 и 3 и их результат на рисунках 2-4.



**Рис. 2 – Ввод числа 2**

Функции, как и в любом другом простом калькуляторе, не запрограммированы быть показанными в строке действия, поэтому их нельзя увидеть – это является упущение данного приложения и может быть исправлено в следующих его версиях.



**Рис. 3 – Ввод числа 3**



**Рис. 4 – Результат**

**Заключение**

В данной выпускной квалификационной работе было разработано приложение «Калькулятор» на основе объектно-ориентированного подхода. Этот метод позволяет обеспечить разделение данных, управление доступом к ним, а также расширяемость и модифицируемость.

В проекте удалось реализовать все поставленные задачи:

1. Создать удобный и понятный графический интерфейс;
2. Создать сохранение состояния приложения во время его работы;
3. Реализовать возможность ввода и удаления данных.

Таким образом, реализовано приложение «Калькулятор», которое позволяет ученику начальной школы получить решение несложных арифметических задач с целью усвоения знаний школьной программы, а также изучения базовых возможностей калькулятора для пользования более сложными его вариантами в будущем.

Для развития приложения в последующих его версиях можно добавить выполнение функций, изучаемых на следующих этапах школьной программы.

**Список использованной литературы**

1. СДО «MOODLE», курс «Разработчик цифровых продуктов»;
2. <https://vertex-academy.com/tutorials/ru/samouchitel-po-java-s-nulya/>;
3. <https://metanit.com/java/tutorial/>;
4. <https://java9.ru/>.

**Приложени****е**

Код класса *NewJFrame*:

public class NewJFrame extends javax.swing.JFrame {

/\*\*

\* Creates new form NewJFrame

\*/

public NewJFrame() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

private void formWindowActivated(java.awt.event.WindowEvent evt) {

// TODO add your handling code here:

Font fn = new Font(Font.MONOSPACED, Font.ITALIC + Font.PLAIN, 14);

Font fn1 = new Font(Font.MONOSPACED, Font.ITALIC + Font.BOLD, 14);

jButton\_one.setFont(fn);

jButton\_two.setFont(fn);

jButton\_three.setFont(fn);

jButton\_four.setFont(fn);

jButton\_five.setFont(fn);

jButton\_six.setFont(fn);

jButton\_seven.setFont(fn);

jButton\_eight.setFont(fn);

jButton\_nine.setFont(fn);

jButton\_zero.setFont(fn);

jButton\_clean.setFont(fn);

jButton\_equals.setFont(fn);

jButton\_plus.setFont(fn1);

jButton\_minus.setFont(fn1);

jButton\_multipy.setFont(fn1);

jButton\_divide.setFont(fn1);

}

private void jButton\_oneActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("1");

}

private void jButton\_twoActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("2");

}

private void jButton\_threeActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("3");

}

private void jButton\_plusActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

firstDigit = Integer.parseInt(EntryField.getText());

EntryField.setText("0");

deistEnum = Deist.PLUSC;

}

private void jButton\_minusActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

firstDigit = Integer.parseInt(EntryField.getText());

EntryField.setText("0");

deistEnum = Deist.MINUSC;

}

private void jButton\_sixActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("6");

}

private void jButton\_fiveActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("5");

}

private void jButton\_fourActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("4");

}

private void jButton\_sevenActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("7");

}

private void jButton\_eightActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("8");

}

private void jButton\_nineActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

symbolDisplay("9");

}

private void jButton\_multipyActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

firstDigit = Integer.parseInt(EntryField.getText());

EntryField.setText("0");

deistEnum = Deist.MULTIPLYC;

}

private void jButton\_divideActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

firstDigit = Integer.parseInt(EntryField.getText());

EntryField.setText("0");

deistEnum = Deist.DIVIDEC;

}

private void jButton\_equalsActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

switch (deistEnum) {

case PLUSC:

EntryField.setText("" + (firstDigit + Integer.parseInt(EntryField.getText())));

break;

case MINUSC:

EntryField.setText("" + (firstDigit - Integer.parseInt(EntryField.getText())));

break;

case MULTIPLYC:

EntryField.setText("" + (firstDigit \* Integer.parseInt(EntryField.getText())));

break;

case DIVIDEC:

EntryField.setText("" + (firstDigit / Float.parseFloat(EntryField.getText())));

break;

default: {

System.out.println("Что-то пошло не так");

}

}

}

private void jButton\_cleanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

EntryField.setText("0");

firstDigit = 0;

}