Содержание

[Введение 3](#_Toc533207633)

[1. Анализ объекта 4](#_Toc533207634)

[1.1 Описание предметной области. 4](#_Toc533207635)

[1.2 Построение концептуальной модели предметной области. 5](#_Toc533207636)

[2 Постановка задачи 6](#_Toc533207637)

[2.1 Определение требований к программной системе. 6](#_Toc533207638)

[2.2 Описание аналогов системы. 7](#_Toc533207639)

[2.3 Обзор и обоснование выбора средств реализации 8](#_Toc533207640)

[3 Проектирование 9](#_Toc533207641)

[3.1 Разработка архитектуры программного продукта 9](#_Toc533207642)

[3.2 Проектирование структур хранения данных. 10](#_Toc533207643)

[4 Реализация 11](#_Toc533207644)

[4.1 Разработка архитектуры программного продукта 11](#_Toc533207645)

[4.2 Разработка интерфейса программного продукта 13](#_Toc533207646)

[4.3 Разработка алгоритмов реализации вариантов использования. 18](#_Toc533207647)

[5 Тестирование 25](#_Toc533207648)

[Литература 29](#_Toc533207649)

[Приложение А 30](#_Toc533207650)

# Введение

Численные методы – достаточно большой раздел высшей математики и серьезные учебники по данной теме насчитывают сотни страниц. На практике, в контрольных работах традиционно предлагаются для решения некоторые задачи по численным методам, и одной из распространенных задач является – приближенное вычисление определенных интегралов. Моя программа позволяет определять приближенного вычисления определенного интеграла.

Данная курсовая работа выполнена на языке программирования высокого уровня Java с использованием IDE Eclipse.

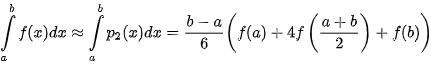
# Анализ объекта

## Описание предметной области.

Численное интегрирование — вычисление значения определённого интеграла (как правило, приближённое). Численное интегрирование применяется, когда:

1. Сама подынтегральная функция не задана аналитически. Например, она представлена в виде таблицы (массива) значений в узлах некоторой расчётной сетки.
2. Аналитическое представление подынтегральной функции известно, но её первообразная не выражается через аналитические функции.

Формулой Симпсона называется интеграл от интерполяционного многочлена второй степени на отрезке [a,b]:



## Построение концептуальной модели предметной области.

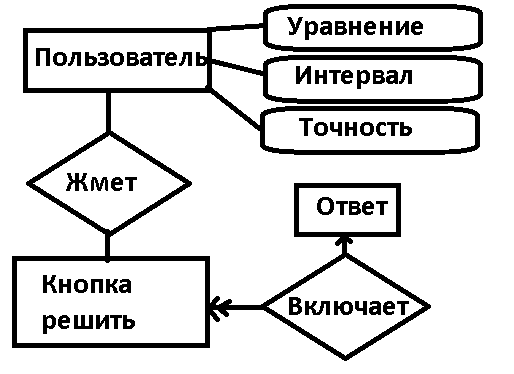


Рисунок 1.2 – модель предметной области

Концептуальная модель состоит из уравнения, которое предстоит решить, его пределов и точности решения. Когда пользователь нажимает кнопку решить, его значения подставляются в формулу Симпсона и уравнение решается. В поле ответа выводится решение с заданной точностью.

# Постановка задачи

## Определение требований к программной системе.

1. Приложение, позволяющее решить уравнение методом Симпсона
2. Поддержка ввода данных как с экрана, так и с файла
3. Поддержка вывода данных как на экран, так и в файл
4. Дружелюбный интерфейс
5. Поддержка Java версии 9 и более

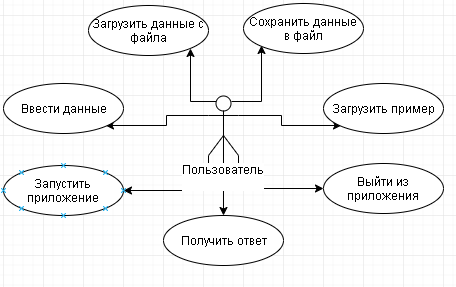


Рисунок 2.1 - диаграмма вариантов использования

## Описание аналогов системы.

В сети Интернет можно найти большое количество работ, которые реализуют решение данных уравнений. Практически все они представляют собой сайты, так что для решения уравнений необходимо подключение к интернету. Примером таким сайтов являются «math.semestr.ru» и «planetcalc.ru».

Мой курсовой проект представляет собой приложение для ПК. Программа не требует подключения к интернету и имеет функции загрузки и выгрузки данных в файл.

### 2.3 Обзор и обоснование выбора средств реализации

Eclipse —интегрированная среда разработки ПО. Написана на языке Java в 2003 году. Также является кроссплатформенной. За счёт присоединяемых к этой среде дополнений — имеется возможность создавать программные продукты более чем на пяти языках программного кода.

Достоинства: Постоянное обновление версий среды разработки, поддержка многих языков (в том числе и русского), является бесплатной, поддержка многих языков программирования, среда имеет промышленный уровень, является гибкой — то есть легко настраивается как под любую платформу, так и под любого пользователя.

Для создания интерфейса было применено:

JFrame – класс, реализующий отображение окон приложения.

JLabel – элементы отображения надписей

JButton – элементы отображения кнопок

JTextField – элементы отображение полей ввода

Для вычисления значений форму было применено:

mXparser – библиотека, позволяющая интерпретировать строку как уравнение и подставлять значения вместо переменых.

# Проектирование

## Разработка архитектуры программного продукта

Поскольку приложение не сетевое, для лучшей скорости сделано однопоточным.

Главное окно будет содержать поля ввода формулы, диапазона измерений, точности измерений. Будет кнопки:

«Решить» - для решения уравнения

«Пример» - пример ввода данных в ручную

«Помощь» - пример ввода данных с файла

«Загрузить» - загрузка данных с файла

«Сохранить» - охранение данных в файл

«Очистить» - очищение всех полей.

Также будет поле вывода ответа. После нажатия кнопки решения, все данные передаются в метод и подставляются в формулу. После вычисления, ответ записывается в поле вывода и показывается пользователю.

## Проектирование структур хранения данных.

Внутреннее хранение данных не предусмотрено. Стандартный пример, который доступен пользователю вшит в программу для предотвращения его потери пользователем. Для хранения данных извне будет использован текстовый файл, данные в нем будут храниться построчно: точность, интервал от, интервал до, функция.

# Реализация

## Разработка архитектуры программного продукта

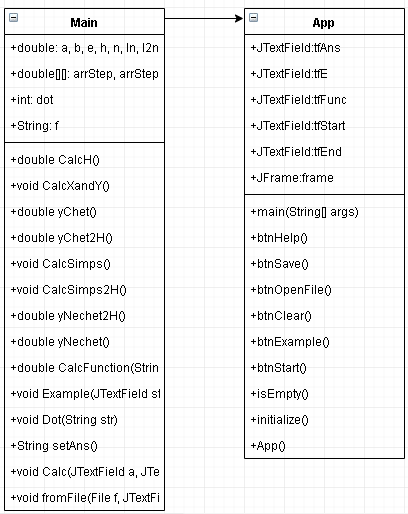


Рисунок 4.1 – Диаграмма классов

Главным классом является класс App, у которого есть переменные для ввода функции, предела, диапазона, точности и вывода

В форме App, при нажатии кнопки решения, происходит проверка на нулевые значение. Если не все данные введены или введены ошибочные – пользователь получит предупреждающее окно. При правильных данных классу Main передаются значения пользователя, которые он обрабатывает и возвращает.

## Разработка интерфейса программного продукта

Главное окно

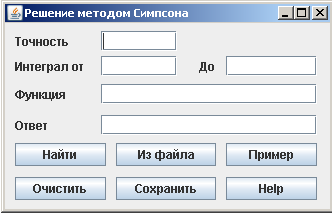


Рисунок 4.2.1 – Вид главного окна

При попытке найти решение без заполнения данныъ, будет показано соответствующее предупреждение.

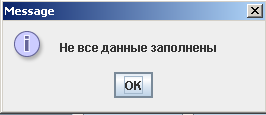


Рисунок 4.2.2 – Ошибка при не заполненных полях

Нажатие на кнопку примера заполнит поля данными.

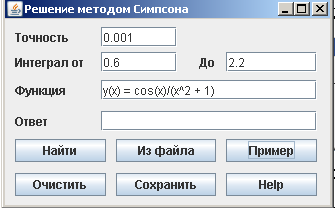


Рисунок 4.2.3 – Поля, заполненные данными из примера

При нажатии на кнопку Help пользователю покажет дополнительную информацию по заполнению файла для выгрузки данных

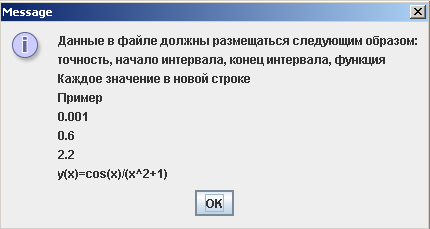


Рисунок 4.2.4 –Информация о загружаемых файлах.

Нажатие нопки «Найти» выводит в поле ответ

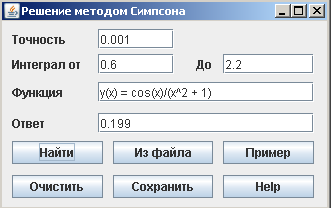


Рисунок 4.2.5 – Форма с выведенным ответом

При нажатии кнопки сохранить, будет выведено окно с выбором папки и имени файла для сохранения.

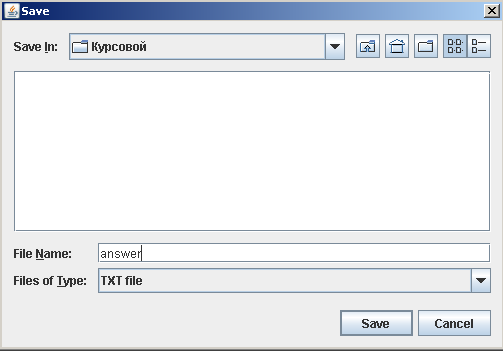


Рисунок 4.2.6 – Окно сохранения файла

При успешном сохранении пользователю выведется соответствующее уведомление

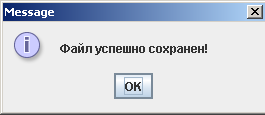


Рисунок 4.2.7 – Уведомление о сохранении

Сохраненный файл выглядит следующим образом

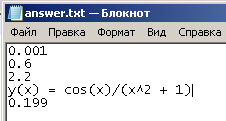


Рисунок 4.2.8 – Вид сохраненного файла

Для примера загрузки файла используем следующие данные

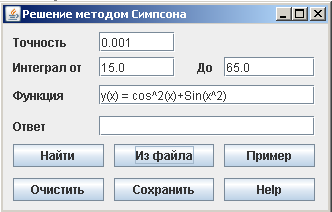
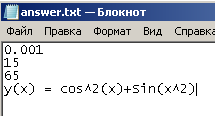


Рисунок 4.2.9 – Успешная загрузка из файла.

## Разработка алгоритмов реализации вариантов использования.

Общий алгоритм работы приложения:

1. Открывается главное меню, переход к 2
2. Вводит точность, если еще не введена, переход к 3
3. Вводит интеграл от, если еще не введен, переход к 4
4. Вводит интеграл до, если еще не введен, переход к 5
5. Ввод функции, если еще не введена, переход к 6
6. Если нажата кнопка Help, вывод уведомления, переход к 7
7. Если нажата кнопка Пример, заполнение всех полей, переход к 8
8. Если нажата кнопка из фала, вывод окна выбора файла, переход к 9
9. Если нажата кнопка сохранить, вывод окна выбора места сохранения файла, переход к 10
10. Если нажата кнопка найти, проверка на заполнение всех полей. Если поля заполнены – переход к 11, если хоть одно не заполнено – переход к2
11. Вычисление функции, вывод ответа, переход к 12
12. Если нажата кнопка очистить – очистка всех полей, переход к 13
13. Если нажата кнопка выход – переход к 14, иначе к 2
14. Завершение работы приложения
15. Проверка полей на заполненность

boolean isEmpty() {

if (tfStart.getText().length() < 1 || tfEnd.getText().length() < 1 || tfE.getText().length() < 1

|| tfFunc.getText().length() < 1)

return true;

else

return false;

}

1. Кнопка открыть файл

void btnOpenFile() {

JFileChooser fileopen = new JFileChooser();

FileFilter filter = new FileNameExtensionFilter("TXT file", new String[] { "txt" });

fileopen.setAcceptAllFileFilterUsed(false);

fileopen.setFileFilter(filter);

int ret = fileopen.showDialog(null, "Выберите файл");

if (ret == JFileChooser.APPROVE\_OPTION) {

File file = fileopen.getSelectedFile();

main.fromFile(file, tfStart, tfEnd, tfFunc, tfE, tfAns);

}

}

1. Вычисление значения методом Симпсона

void CalcSimps() {

In = (h / 3) \* (arrStep[0][1] + arrStep[(int) n][1] + 4 \* yNechet() + 2 \* yChet());

}

1. Вспомогательные методы, находящий сумму нечетных и четных элементов

double yNechet() {

double sum = 0;

for (int i = 1; i < arrStep.length - 1; i++) {

if (i % 2 != 0)sum += arrStep[i][1];}

return sum; }

double yChet() {

double sum = 0;

for (int i = 1; i < arrStep.length - 1; i++) {

if (i % 2 == 0)sum += arrStep[i][1]; }

return sum; }

# Тестирование

Проведем тестирование программы, пройдя все варианты взаимодействия. Запустим само приложение. Запуск происходит без проблем.

Попробуем найти решение не вводя значения, затем по примеру и из файла

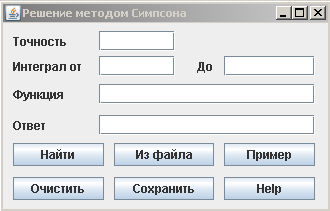


Рисунок 5.1 – Запуск главного окна

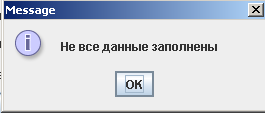


Рисунок 5.2 – Попытка запуска игры без введенных значений

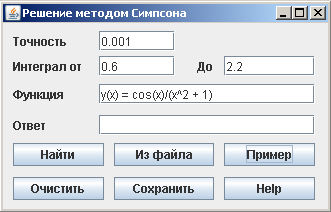


Рисунок 5.3 – Работа кнопки примера

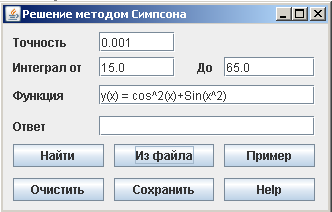
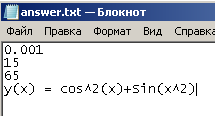


Рисунок 5.4 – Данные из файла

Проверяем работу кнопки Найти

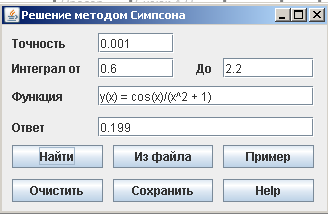


Рисунок 5.5 – Нахождение ответа

Работа приложения происходит без ошибок

## Литература

1. Хорстманн, К.С. Java 2. Библиотека профессионала, том1. Основы. 8-е издание, : Пер. с англ. / К. С. Хорстманн, Г. Корнелл - М.: ООО «Вильямс», 2012. - 816 с.
2. Дирк, Л. Самоучитель Java 7: Пер. с нем. / Л. Дирк, П. Мюллер. – СПб: БХВ-Петербург, 2013. – 464 с.: илл.
3. Шилдт, Г. Java. Полное руководство. 8-е издание, : Пер. с англ. / Г. Шилдт. – М.:ООО «Вильямс», 2012. – 1104 с.
4. Блинов, И.Н. Java. Промышленное программирование./ И.Н. Блинов, В.С. Романчик –Минск: «Четыре четверти», 2013. – 896 с.

## Приложение А

Исходный код, откомпилированная программа и записка находятся на диске.