МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» (Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра технической кибернетики

**Интегральный отчет по лабораторным работам  
по курсу**

**«Объектно-ориентированное программирование»**

Студент: Борисов Дмитрий Сергеевич  
группа 6210  
Преподаватель: Хотилин М.И.

Самара 2018

Содержание

[ВВЕДЕНИЕ 3](#_Toc531767132)

[ЗАДАНИЕ 1 4](#_Toc531767133)

[ЗАДАНИЕ 2 4](#_Toc531767134)

[ЗАДАНИЕ 3 4](#_Toc531767135)

[ЗАДАНИЕ 4 5](#_Toc531767136)

[ЗАДАНИЕ 5 5](#_Toc531767137)

[ЗАДАНИЕ 6 5](#_Toc531767138)

[ЗАДАНИЕ 7 6](#_Toc531767139)

[ЗАДАНИЕ 8 6](#_Toc531767140)

[ЗАКЛЮЧЕНИЕ 7](#_Toc531767141)

**ВВЕДЕНИЕ**

В рамках курса лабораторных работ, состоящего из восьми этапов (по этапу на каждое лабораторное занятие) нужно ознакомиться со структурой исходного кода для Java, а также изучить и апробировать особенности областей видимости и использования пакетов, лексики и базовых конструкций языка, использования интерфейсов и обработки исключительных ситуаций , использования потоков ввода/вывода, в том числе состояния объекта (сериализации /десериализации), создания, перегрузки и переопределения методов (в том числе класса Object), создания и использования конструкций, определенных паттернами проектирования. Необходимый программный функционал должен быть реализован в виде графического приложения со стандартным набором компонент, для расчета ряда результатов должен быть применен многопоточный подход.

**ЗАДАНИЕ 1**

Было осуществлено написание программного кода, состоящего из двух классов, содержащихся в разных пакетах, были изучены особенности областей видимости, использования пакетов и создания jar-архивов.

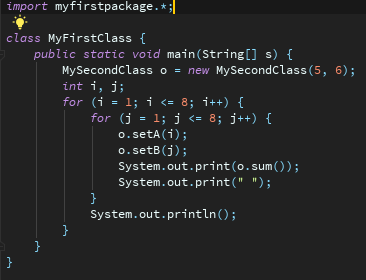
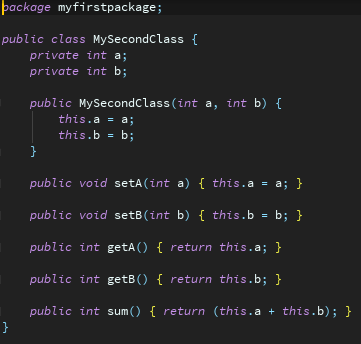
 

Рисунок 1 Рисунок 2

**ЗАДАНИЕ 2**

Был создан набор классов для работы с функциями одной переменной, заданными в табличной форме. (Ниже приведены лишь части классов, подробнее на сайте: <https://github.com/DiBoS290699/Labs/tree/master/OOP_2> )

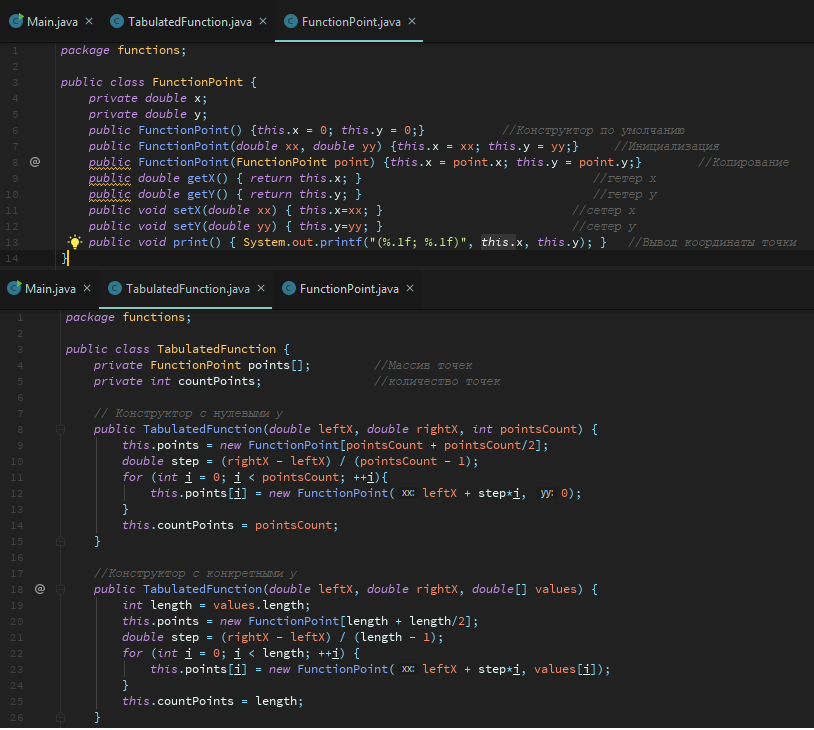


Рисунок 3

**ЗАДАНИЕ 3**

Созданный в задании 2 пакет для работы с функциями одной переменной, заданными в табличной форме, был дополнен классами исключений, классом функций и базовым интерфейсом. (Класс функций на сайте GitHub: <https://github.com/DiBoS290699/Labs/tree/master/OOP_3/src> )

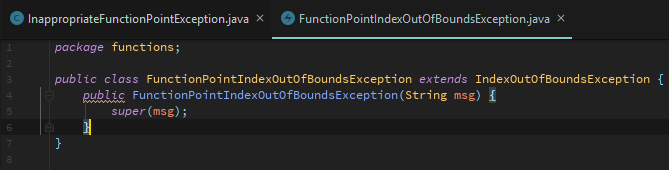
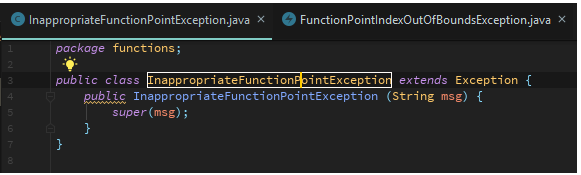


Рисунок 4



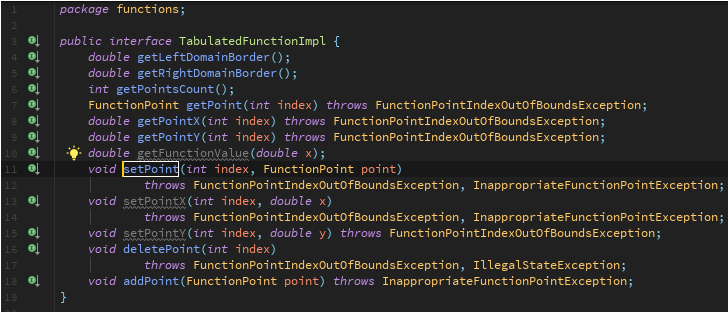
Рисунок 5

Рисунок 6

**ЗАДАНИЕ 4**

В проект были добавлены интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций, а также добавлена возможность сериализации и десериализации объектов. (Ниже приведены примеры этих классов, остальные классы можно найти на сайте: <https://github.com/DiBoS290699/Labs/tree/master/OOP_4> )

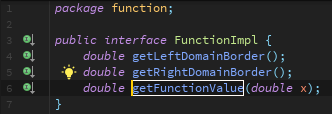
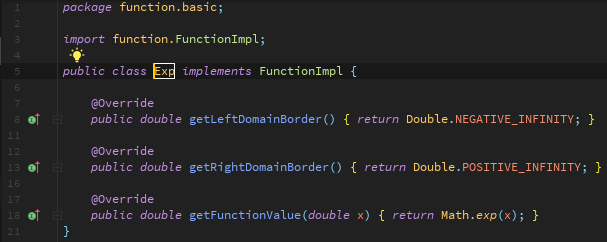


Рисунок 7



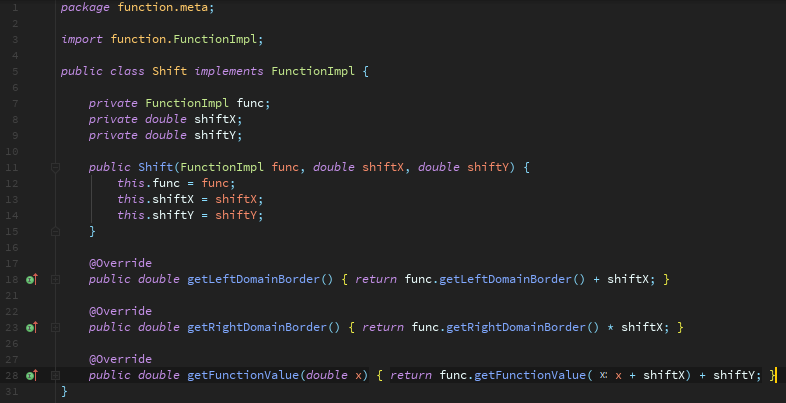
Рисунок 8

Рисунок 9

**ЗАДАНИЕ 5**

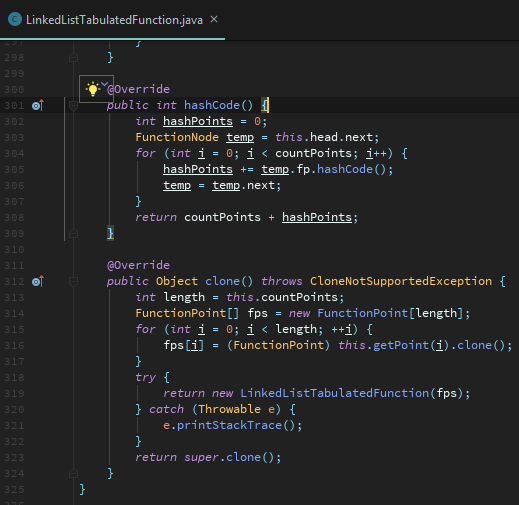
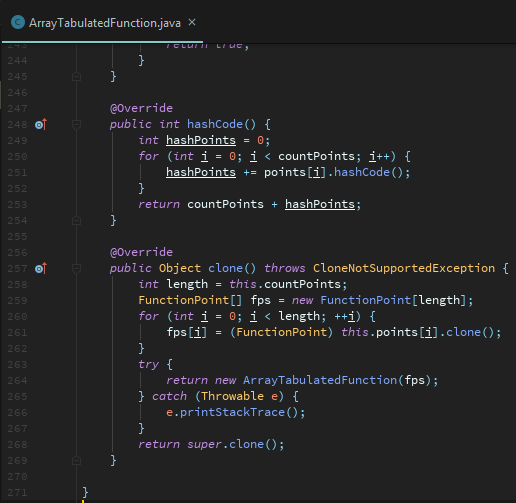
Расширены возможности классов, связанных с табулированными функциями за счет переопределения методов, унаследованных из класса Object. (Ниже приведены некоторые примеры этих методов, другие можно найти тут: <https://github.com/DiBoS290699/Labs/tree/master/OOP_5/src> )

Рисунок 10 Рисунок 11

**ЗАДАНИЕ 6**

Под действующий функционал было создано графическое приложение на платформе SWING. Был реализован класс вспомогательного окна, в котором вводятся параметры табулирования функции (Рисунок 12), класс документа с табулированной функцией (Рисунок 13), класс модели для таблицы, содержащий конструктор и ряд методов (рисунок 14), класс основного окна программы и его функционал (Рисунок 15-19), класс загрузчика классов из указанного файла (Рисунок 20). Ниже приведены скриншоты лишь небольшой части классов, подробнее на сайте: <https://github.com/DiBoS290699/Labs/tree/master/OOP_6/src> )

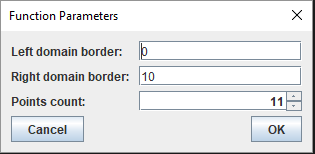


Рисунок 12

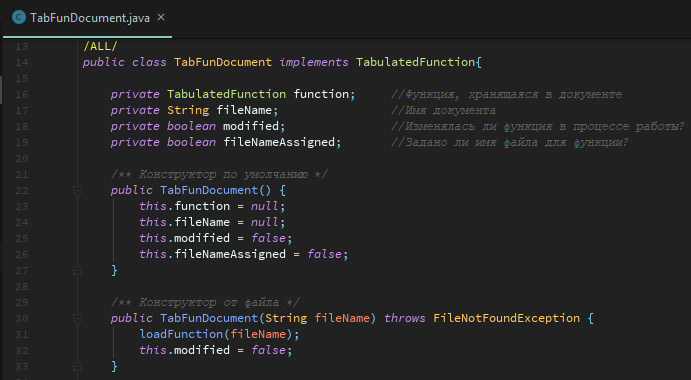


Рисунок 13

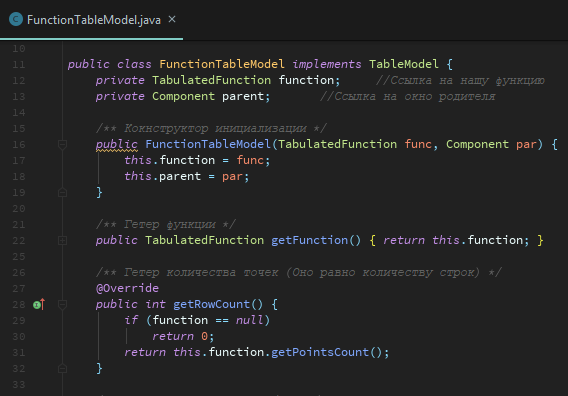


Рисунок 14

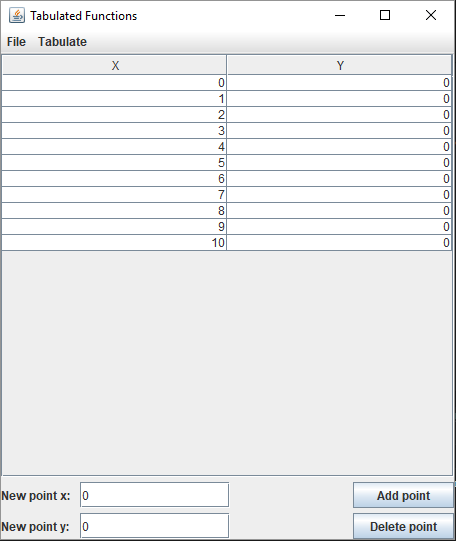
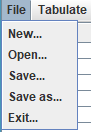


Рисунок 16



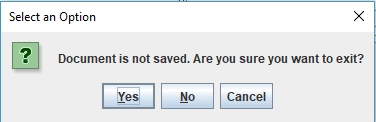
Рисунок 17

Рисунок 18

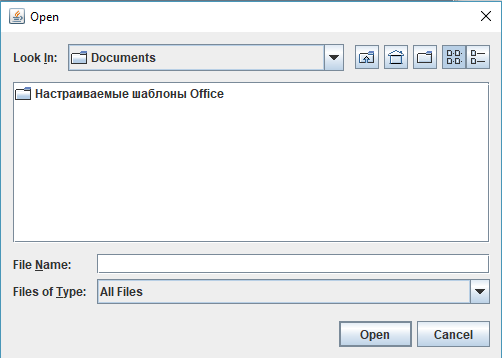


Рисунок 19

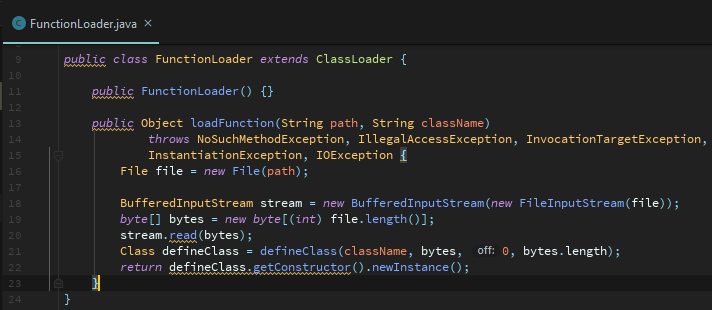


Рисунок 20

**ЗАДАНИЕ 7**

В приложение добавлен функционал многопоточных вычислений. В одном потоке вычислений формируется набор заданий для интегрирования (Рисунок 21), а во втором потоке – вычисляющее значения интегралов функций (Рисунок 22). Ниже приведены примеры генератора и интегратора с методом синхронизации потоков, аналогичные классы, но с использованием семафора можно найти на сайте: <https://github.com/DiBoS290699/Labs/tree/master/OOP_7/src/function/threads> )

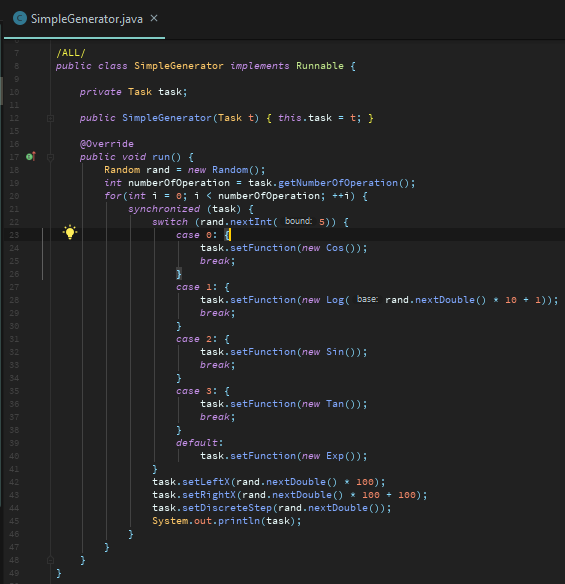


Рисунок 21

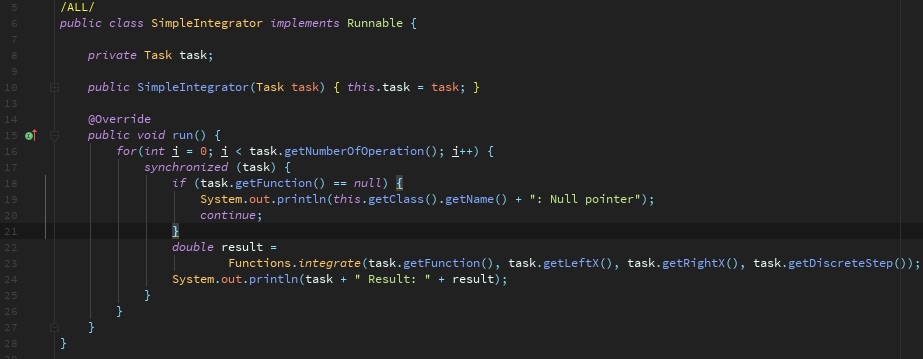


Рисунок 22

**ЗАДАНИЕ 8**

Внесены изменения в существующий набор типов табулированных функций, позволяющие обрабатывать точки функций по порядку (паттерн «Итератор»), а также выбирать тип объекта табулированной функции при его неявном создании (паттерн «Фабричный метод» и средства рефлексии)…

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения заданий произведено ознакомление со структурой исходного кода объектно-ориентированного языка высокого уровня Java, с базовыми классами и стандартной библиотекой. Изучены и апробированы особенности областей видимости и использования пакетов, лексики и базовых конструкций языка, использования интерфейсов и обработки исключительных ситуаций , использования потоков ввода/вывода, в том числе состояния объекта (сериализации /десериализации), создания, перегрузки и переопределения методов (в том числе класса Object), создания и использования конструкций, определенных паттернами проектирования. Необходимый программный функционал реализован в виде графического приложения со стандартным набором компонент. Для вычисления результатов применен многопоточный подход, в том числе регулируемый с помощью объекта семафора.

При написании программного приложения были выполнены следующие требования:

- продемонстрировано владение языком Java, его стандартной библиотекой, для создания нетривиальной реализации необходимых методов и алгоритмов;

- необходимый функционал реализован, основываясь на таких базовых парадигмах как инкапсуляция, наследование и полиморфизм,

- выполнено соответствие принципам SOLID,

- программный код написан в полном соответствии с правилами именования и комментариями, в том числе с применением технологии Javadoc.