Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

Технологий»

ОТЧЕТ

по учебной ознакомительной практике

Исполнитель

студент 1 курса 4 группы \_\_\_\_\_\_\_\_\_\_\_\_\_ Корнелюк В. В

(подпись, дата) (Ф.И.О.)

Руководитель практики

от университета

Преп.-стажер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Север А. С

(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2023 г.

**06.02.2023.** Была просмотрена лекция от EPAM на тему «Открытие IT Warm-Up» в которой были рассмотрены следующие моменты:

- руководство по IT Warm-Up, программа практики;

- презентация EPAM & EPAM Training Center;

- портал training.by.

Домашнее задание было успешно выполнено: был зарегистрирован профиль на сайте training.by (рис. 1.1) и были прочитаны статьи из блога «English Self-study materials». В качестве статей были выбраны 5 квизов, которые были успешно пройдены. И заключительным заданием являлось прохождение теста по английскому языку, который показал текущий уровень знаний (рис. 1.2).

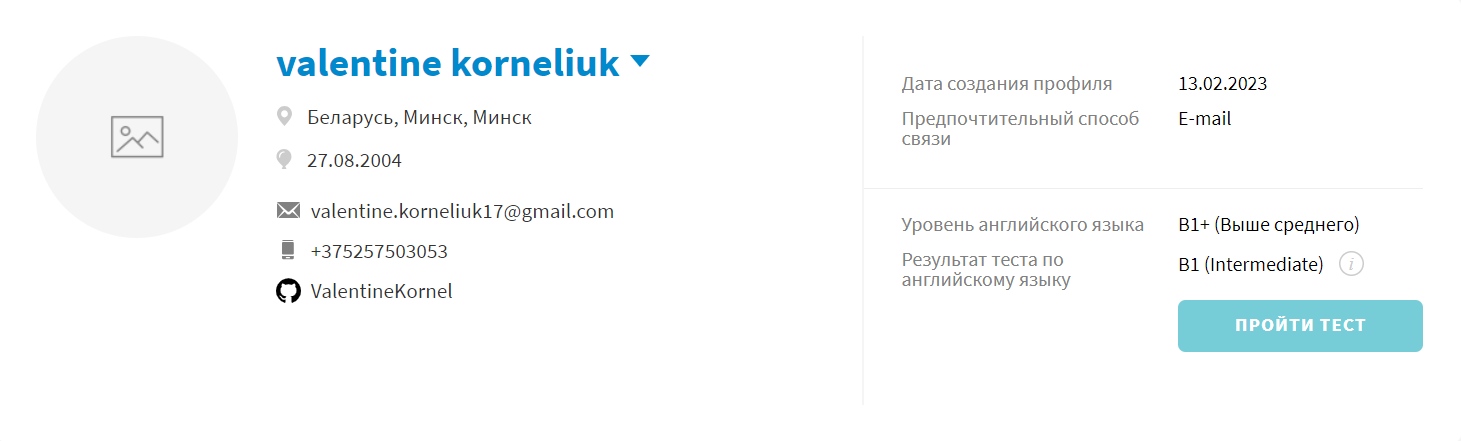


Рисунок 1.1 – Профиль на сайте training.by

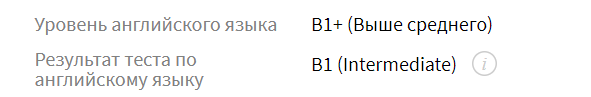


Рисунок 1.2 – Результат теста по английскому языку

**13.02.2023.** Была просмотрена лекция от EPAM на тему «ПОРТАЛ LEARN» в которую вошли следующие моменты:

- портал learn.epam.com;

- курсы Computer Basics, Version Control with Git.

Домашним заданием являлось выполнение программы self-study курсов: «Version Control with Git | Learn» и «Computer Science Basics | Learn». Результаты выполнения курсов представлены на рисунке 1.3.

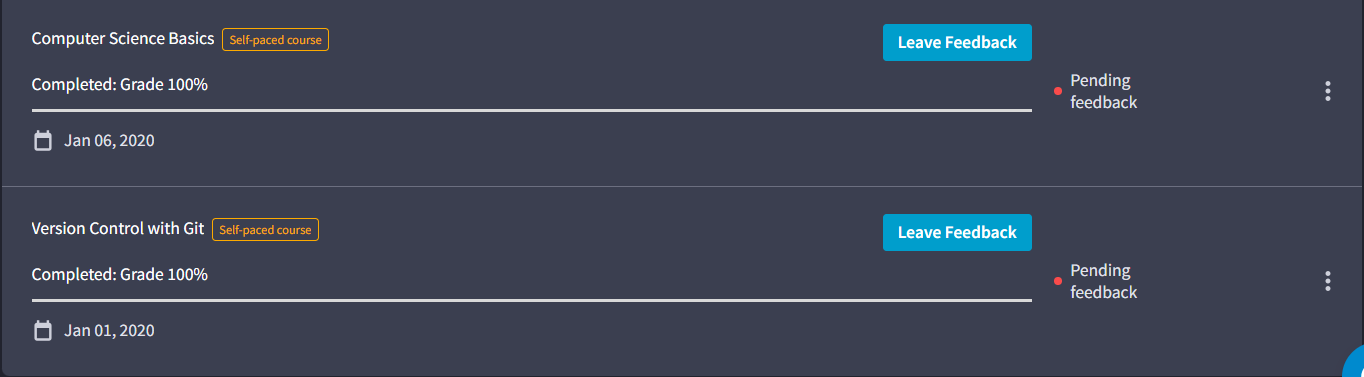


Рисунок 1.3 – Результат выполнения курсов

**20.02.2023.** Была просмотрена лекция от EPAM на тему «ПЛАТФОРМА AUTOCODE & JAVA TECH TALK» в которую вошли следующие моменты:

- проект Autocode;

- взаимодействие инструментов при выполнении заданий;

- Java: презентация направления.

**27.02.2023.** Была просмотрена лекция от EPAM на тему «IT ПРОФЕССИИ & ИНСТРУМЕНТЫ ДЛЯ ОБУЧЕНИЯ» в которую вошли следующие моменты:

- инструменты для обучения;

- погружение в IT & обзор профессий;

Была также просмотрена лекция от EPAM на тему «EPIC INSTITUTE OF TECHNOLOGY» в которую вошли следующие моменты:

- знакомство с EPIC Institute of Technology;

- команда разработчиков-инноваторов DELTIX;

- процесс обучения и обзор программ.

Был успешно пройден тест на платформе Examinator (рис. 1.3).

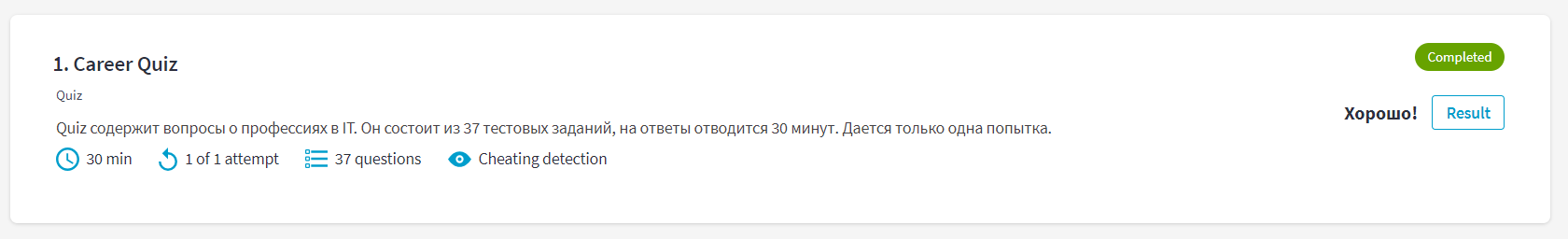


Рисунок 1.3 – Результат теста

**06.03.2023.** Были просмотрены лекции от EPAM на темы «TECH TALK: AUTOMATED TESTING», «TECH TALK: CLOUD & DEVOPS», «TECH TALK: FUNCTIONAL TESTING», «TECH TALK: DATA & ANALYTICS» для углубления знаний в IT-сфере.

Был выбран тренинг «Java Basics» который включает в себя 16 разделов, 14 из которых содержат практические задачи, и финального тестирования. Тренинг был выбран по нескольким причинам. Во-первых, Java является одним из наиболее популярных и востребованных языков программирования. Овладение его основами даст прочный фундамент для дальнейшего изучения и применения в реальных проектах. Во-вторых, тренинг предлагает структурированную программу обучения, которая позволит изучать язык пошагово, начиная с основных концепций и переходя к более сложным темам. Кроме того, тренинг обещает практические задания, которые помогут применить полученные знания на практике.

**13.03.2023.** Было начато выполнение выбранного тренинга. Был рассмотрен раздел «Введение в курс Java Basics» в котором описывался формат проведения тренинга, а также давалась инструкция для выполнения и проверки практических заданий.

Был изучен первый раздел - «Первые программы в Java». В теоретической части раздела были рассмотрены следующие подразделы:

- знакомство с Java. Было получено общее представление и языке программирования Java, его истории и применении, а также что Java является объектно-ориентированным языком и широко используется для разработки приложений, веб-сайтов и мобильных приложений;

- установка JDK. Был изучен процесс установки JDK и настроек, которые нужно выполнить для начала разработки. Результат представлен на рис. 1.4;

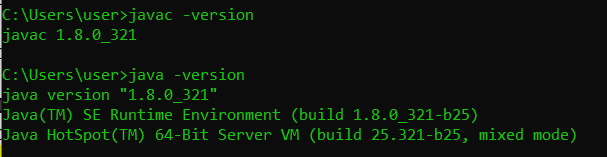


Рисунок 1.4 – Результат установки JDK.

- ключевые слова. Было рассказано про некоторые ключевые слова Java, такие как public, new, void, static, class. Было объяснено их назначение и как они используются при написании программ на Java;

- компиляция и запуск приложения. Консоль. Узнал о процессе компиляции и запуска Java-приложений с помощью командной строки. Изучил, как создавать простые программы, компилировать их в байт-код Java и запускать на выполнение. Создал и скомпилировал с помощью командной строки первое приложение, которое выводило приветственное сообщение (рис 1.5);

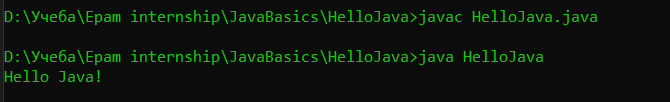


Рисунок 1.5 – Результат компиляции и запуска приложения.

- установка IDE. Изучил процесс установки интегрированной среды разработки IntelliJ IDEA. Узнал о преимуществах использования IDE при разработке, таких как интеграция с системами управления версиями, автодополнение и рефакторинг кода, а также как настроить среду для работы с проектами на Java. Была установлена интегрированная среда разработки IntelliJ IDEA для дальнейшей работы;

- компиляция и запуск приложения IDE. Разобрался в процессе создания, компиляции и запуска Java-приложений с использованием IDE (рис 1.6). Изучил основные функциональные возможности IDE, такие как автодополнение кода, отладка и управление проектами.



Рисунок 1.6 – Результат выполнения приложения через IDE.

Все эти подразделы предоставили мне прочные основы для начала работы с Java. Я приобрел понимание ключевых концепций и инструментов, необходимых для написания и запуска программ на Java. Этот Раздел помог мне сформировать представление о рабочем окружении разработчика Java и подготовил меня к изучению более сложных тем в дальнейшем.

Практическая часть курса была организована следующим образом: Сначала, нужно было зарегистрировать учетную запись на GitLab (рис. 1.7). После чего, используя свою учетную запись GitLab нужно было авторизоваться на платформе Autocode, после чего можно было приступать к выполнению заданий, для этого нужно клонировать репозиторий с заданием, который также включает в себя и тесты для проверки (рис 1.8). После выполнения задания и успешного выполнения тестов, нужно сделать пуш репозитория и отправить его на проверку, и в случае успешного прохождения проверки насчитываются баллы за прохождение задания (рис. 1.9).

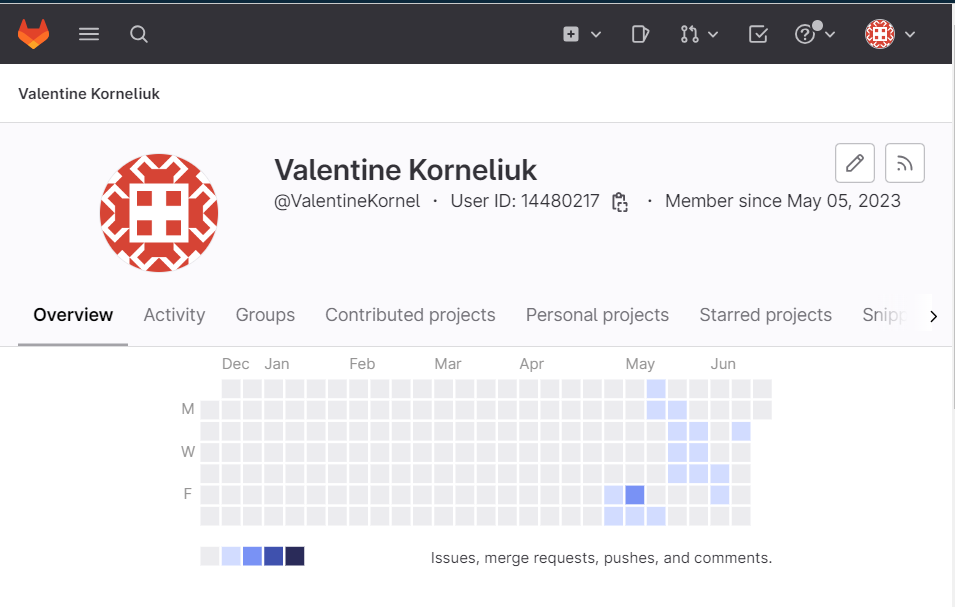


Рисунок 1.7 – Регистрация на GitLab.

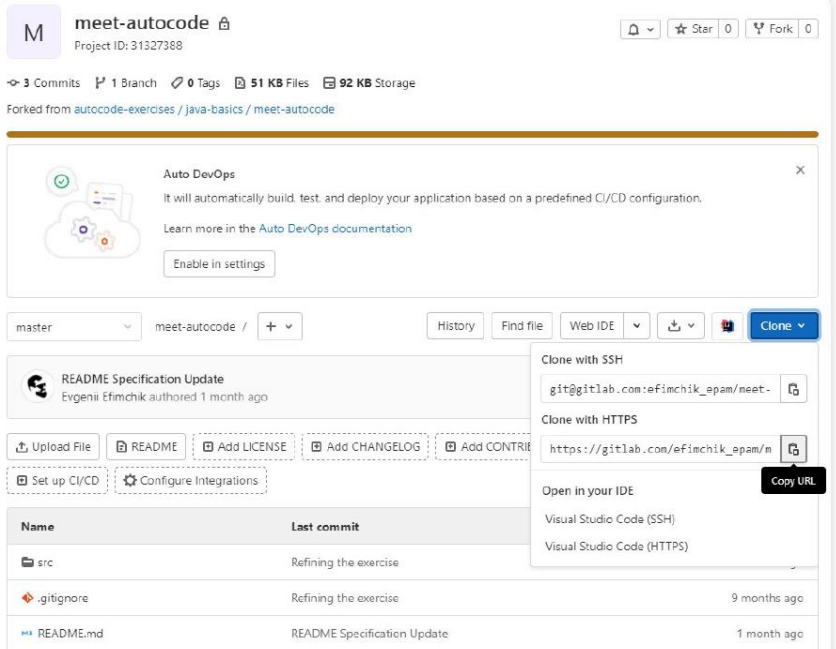


Рисунок 1.8 – Пример репозитория с заданием.

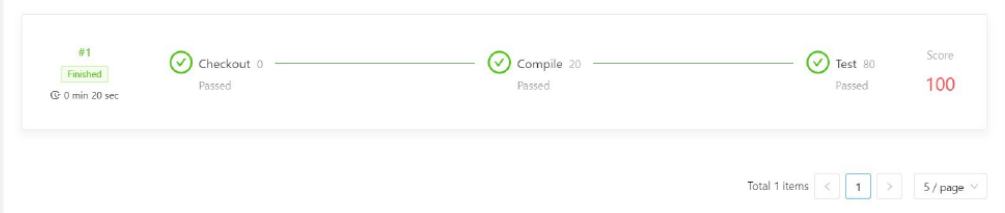


Рисунок 1.9 – Пример проверки задания на платформе Autocode.

В практической части было выполнено 2 предложенных задания:

- “Meet Autocode”. Целью задания было познакомиться с платформой Autocode. Нужно было написать простую программу, которая печатает сообщение “Hello, Autocode”. Результат выполнения задания представлен на рис. 1.10;

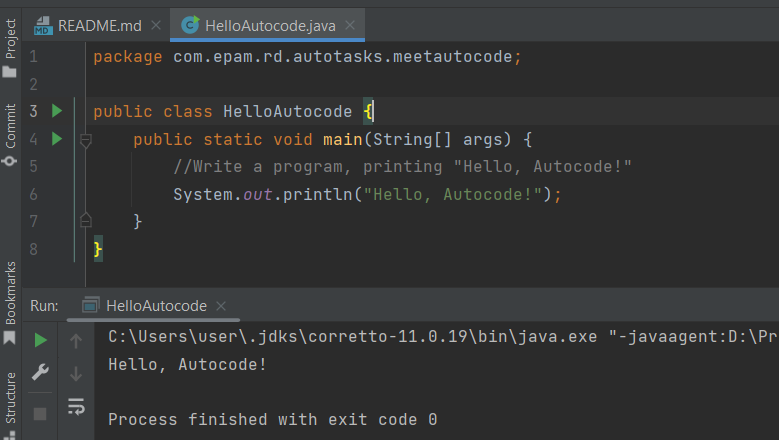


Рисунок 1.10 – Результат выполнения задания “Meet Autocode”.

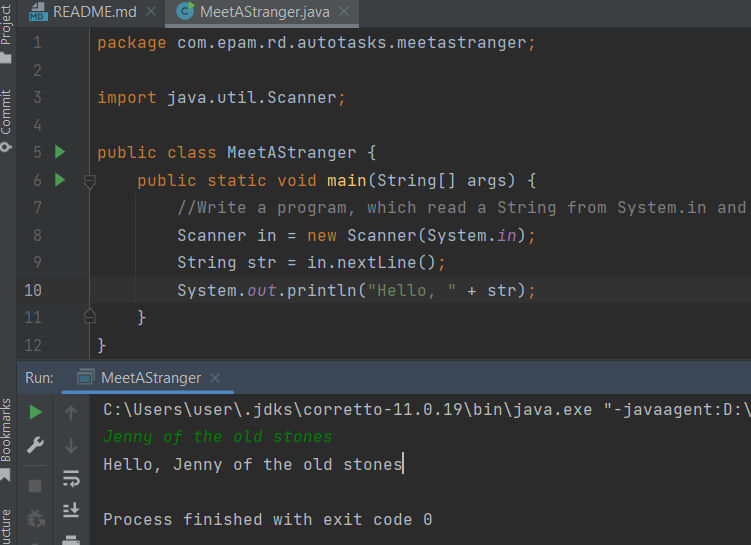
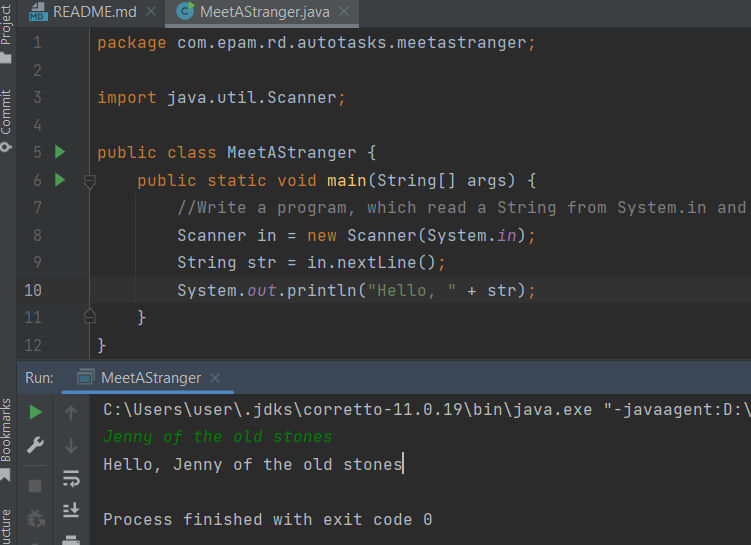
- “Meet a Stranger”. Целью этого задания было познакомится с приемами использования стандартного потока ввода. Нужно было написать программу, которая читает строку с консоли и выдает приветственное сообщение. Результат выполнения задания представлен на рисунке 1.11.

Рисунок 1.11 – Результат выполнения задания “Meet a stranger”.

Был изучен второй раздел – «Типы данных». В теоретической части раздела были рассмотрены следующие подразделы:

- переменные и типы данных в Java. Я изучил основные понятия переменных и типов данных в Java. Узнал, что переменная представляет собой именованную область памяти, которая содержит определенное значение, а тип данных определяет, какую информацию может содержать переменная;

- примитивы. Познакомился с различными примитивными типами данных в Java, такими как целочисленные типы (byte, short, int, long), типы с плавающей точкой (float, double), символьный тип (char) и логический тип (Boolean). Узнал из особенности, диапазоны значений и использование на практике;

- операторы. Изучил различные операторы, которые могут быть применены к примитивным типам данных в Java, включая арифметические операторы, операторы присваивания, операторы сравнения и логические операторы;

- приведение типов. Узнал о понятии приведения типов в Java. Разобрался в необходимости и способах приведения одного типа данных в другому, как явного, так и неявного;

- ссылочные типы. Познакомился со ссылочными типами данных в Java, такими как строки (String) и массивы. Изучил, как объявлять и использовать ссылочные типы, а также некоторые базовые операции с ними;

- конвенция именования в Java. Узнал о важности и правилах именования переменных, классов, методов и других элементов в Java. Изучи рекомендации по именованию и понял, как правильное именование может улучшить читаемость и понимание кода.

Этот раздел помог мне лучше понять различные типы данных, их использование и взаимодействие в Java. Я научился объявлять и использовать переменные различных типов, применять операторы и понимать основы приведения типов. Понятие ссылочных типов и конвенции наименования позволили мне начать разработку более сложных программ и следовать лучшим практикам программирования на Java.

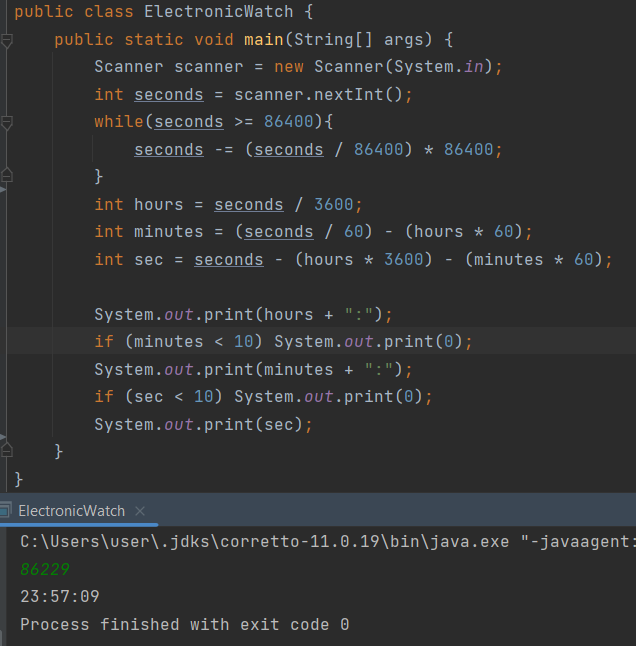
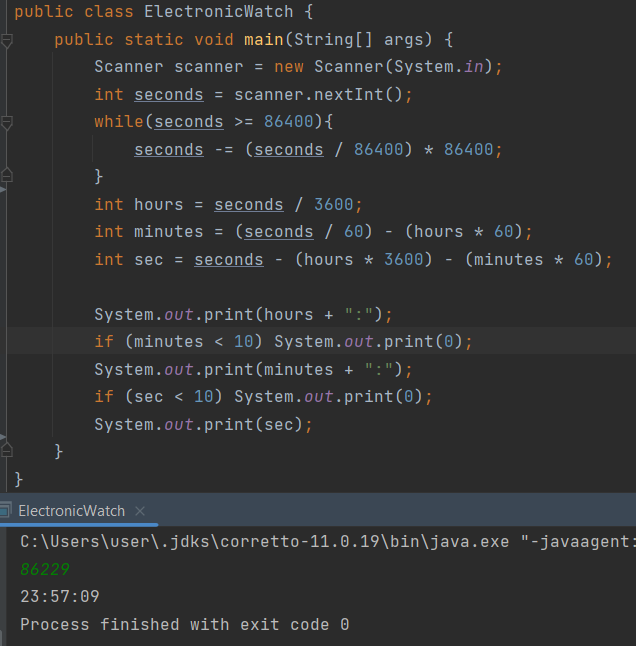
В практической части было выполнено предложенное задание “Electronic Watch”. Целью этого задания было познакомиться с операторами int. Нужно было написать программу, которая получает на вход целое число, которое представляет собой количество прошедших с полуночи секунд. На выходе программа должна напечатать вывод экрана электронных часов для этого времени. Пример выполнения задания представлен на рисунке 1.12.

Рисунок 1.12 – Пример выполнения программы “Electronic Watch”.

**20.03.2023**.Был изучен третий раздел – «Условия и циклы». В теоретической части раздела были следующие подразделы:

- понятие блока кода. Узнал, что блок кода представляет собой группу инструкций, объединенных внутри фигурных скобок. Изучил, как использовать блоки кода для организации и структурирования программы;

- оператор условного перехода. Изучил операторы условного перехода в Java, такие как if, else, if-else. Узнал, как использовать эти операторы для выполнения различных блоков кода в зависимости от заданного условия;

- оператор выбора. Разобрался с оператором выбора switch, который позволяет выбрать один из нескольких вариантов выполнения кода, основываясь на значении определенной переменной или выражения;

- циклы. Изучил различные типы циклов в Java, включая циклы while, do-while и for. Узнал, как использовать эти циклы для многократного выполнения блока кода в зависимости от условий.

Этот раздел позволил мне понять, как структурировать программу с помощью условий и циклов. Я научился использовать операторы условного перехода для принятия решений в зависимости от определенных условий, а также оператор выбора для альтернативного пути выполнения на основе значения переменной или выражения.

В практической части было выполнено 8 предложенных заданий:

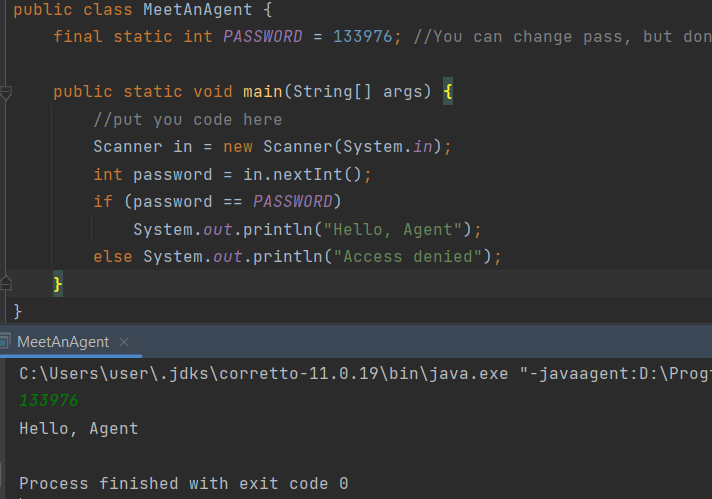
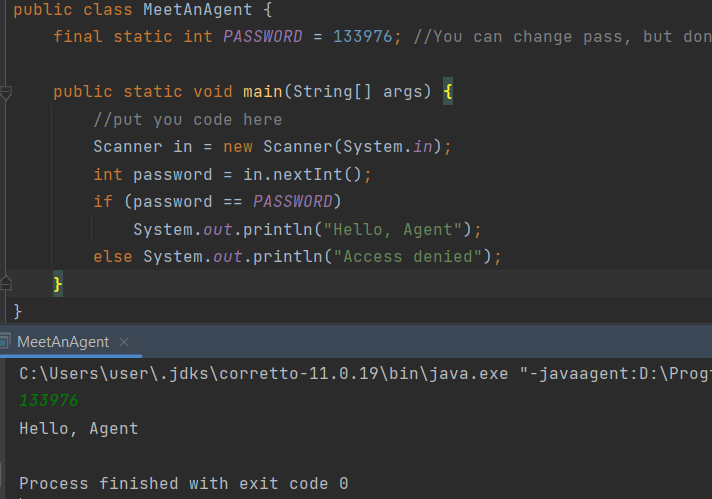
- “Meet an Agent”. Целью этого задания было познакомиться с простыми условными выражениями. Нужно было написать программу, которая запрашивала на вход число-пароль и, если оно равно установленному секретному числу, то программа выводит сообщение “Hello, Agent”, в противном случае – “Access denied”. Пример выполнения программы представлен на рисунке 1.13;

Рисунок 1.13 – Пример выполнения программы “Meet an Agent”.

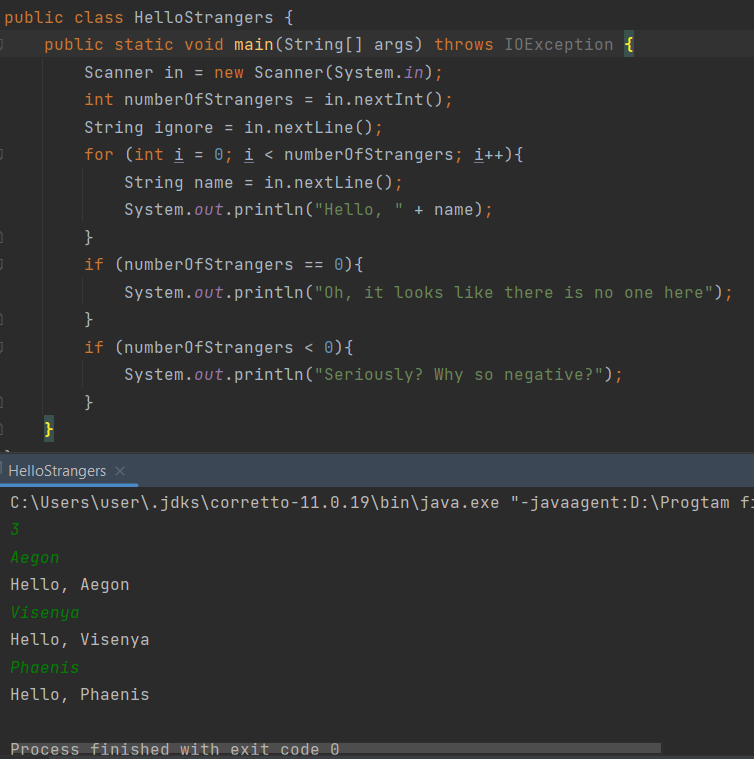
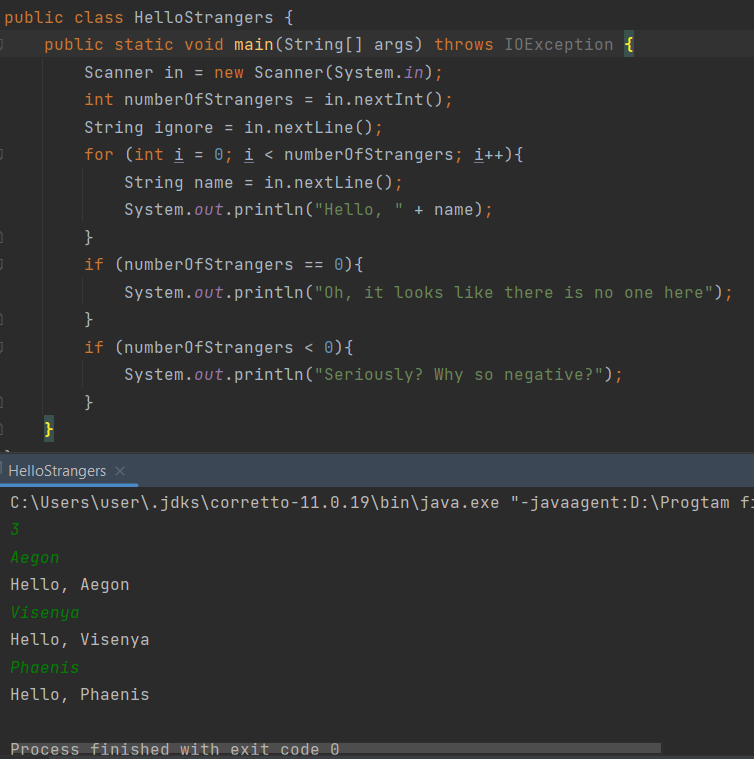
- “Meet Strangers”. Нужно было написать программу, которая запрашивает количество незнакомцев, с которыми нужно встретиться, после чего построчно читает имена незнакомцев и построчно выводит приветствие для каждого незнакомца. Пример выполнения программы представлен на рисунке 1.14;

Рисунок 1.14 – Пример выполнения программы “Meet strangers”.

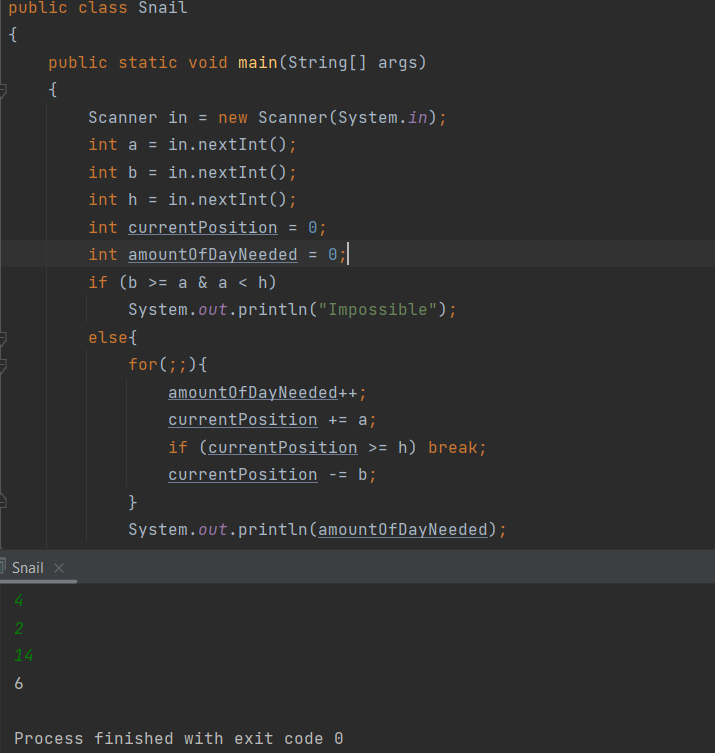
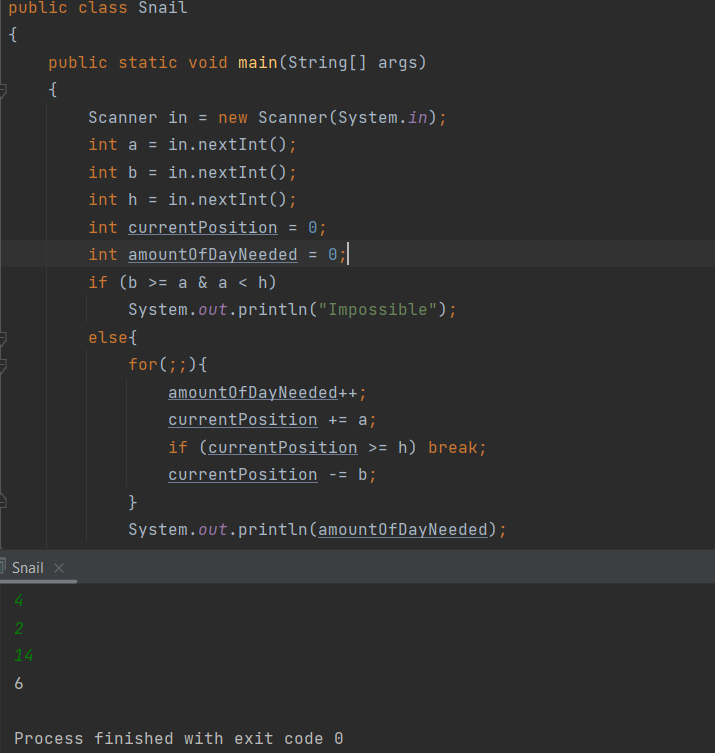
- “Snail”. Целью задания было научиться пользоваться простыми целочисленными операциями. Было дано следующее условие: улитка поднимается по дереву на a футов в день. Затем каждую ночь улитка сползает вниз на b футов. Высота дерева — h футов. Нужно было написать программу с подсчетом количества дней, которые потребуются улитке, чтобы добраться до вершины дерева. Пример выполнения программы представлен на рисунке 1.15;

Рисунок 1.15 – Пример выполнения программы “Snail”.

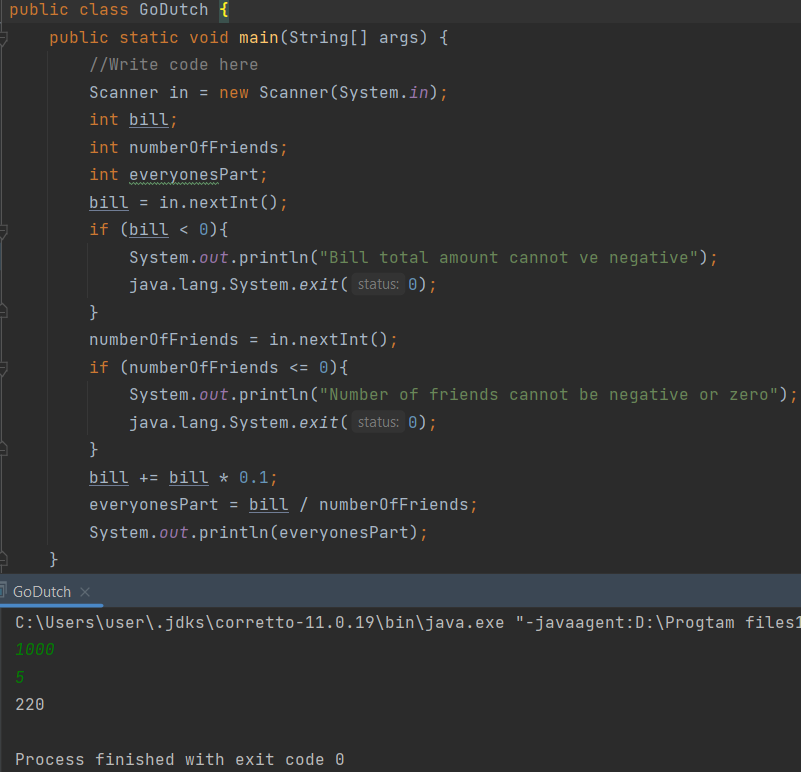
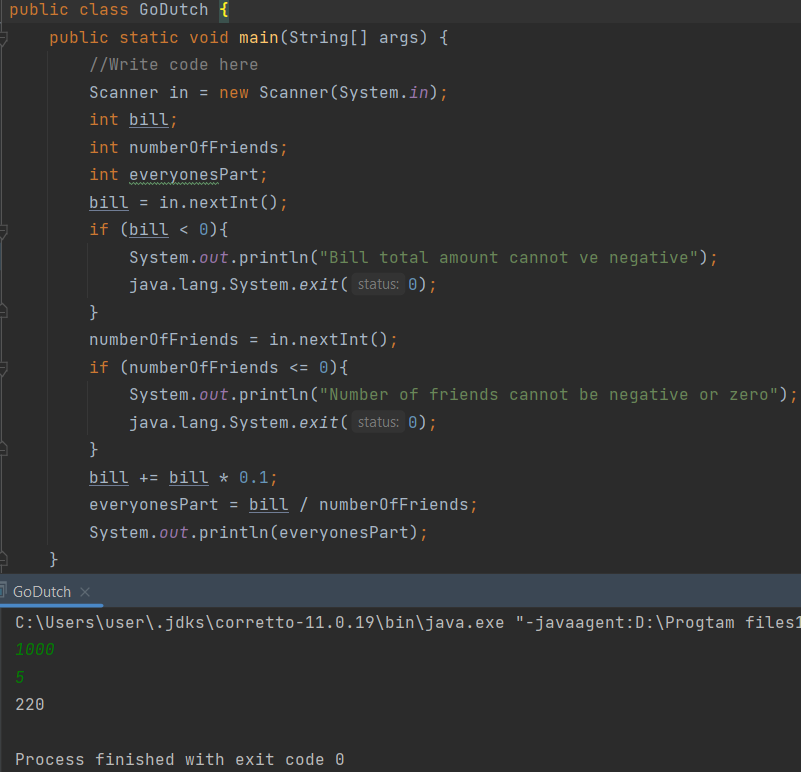
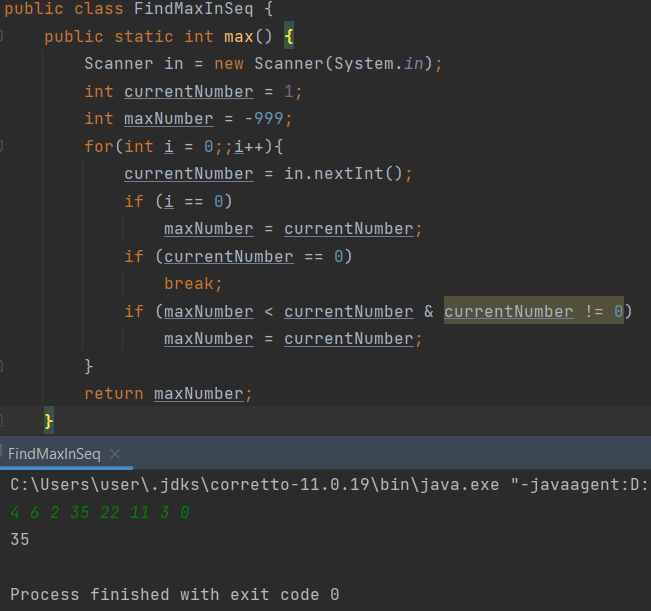
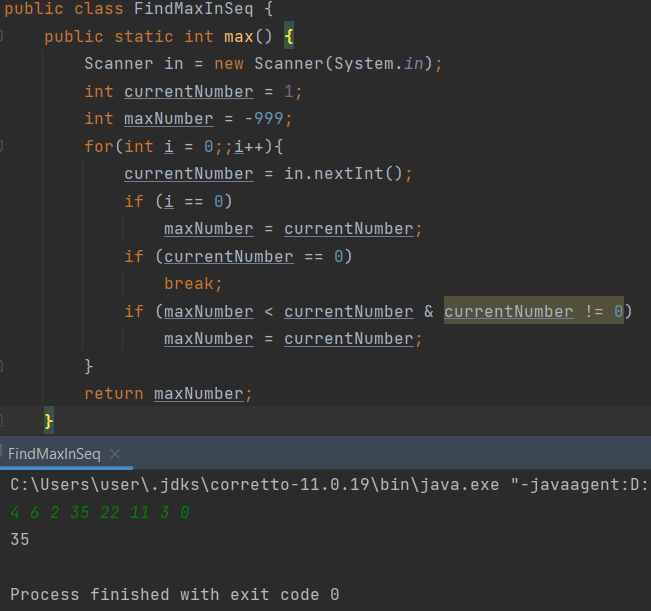
- “Go Dutch”. Целью задания было познакомиться с основными условными и циклическими операциями. Было дано следующее условие: Дано следующее условие: компания друзей посещает ресторан. Они решили разделить счет поровну и добавить 10 процентов от общей суммы счета в качестве чаевых. Далее друзья равными частями покрывают общую сумму платежа. Нужно было написать программу, которая считывает общую сумму счета и количество друзей, а затем выводит размер частит оплаты. Пример выполнения программы представлен на рисунке 1.16.

Рисунок 1.16 – Пример выполнения задания “Go Dutch”.

-“Max Value In Sequeue”. Целью задания было научиться пользоваться простыми циклами и условными опереторами. В этом задании нужно было написать программу, которая считывает последовательность целочисленных значений, заканчивающуюся нулем, из стандартного вывода и находит максимальное из них. Пример выполнения программы представлен на рисунке 1.17;

Рисунок 1.17 – Пример выполнения задания “Max Value In Sequence”.

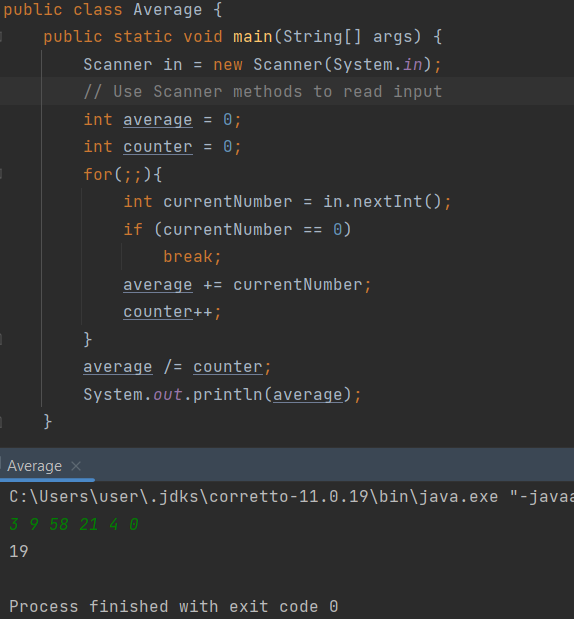
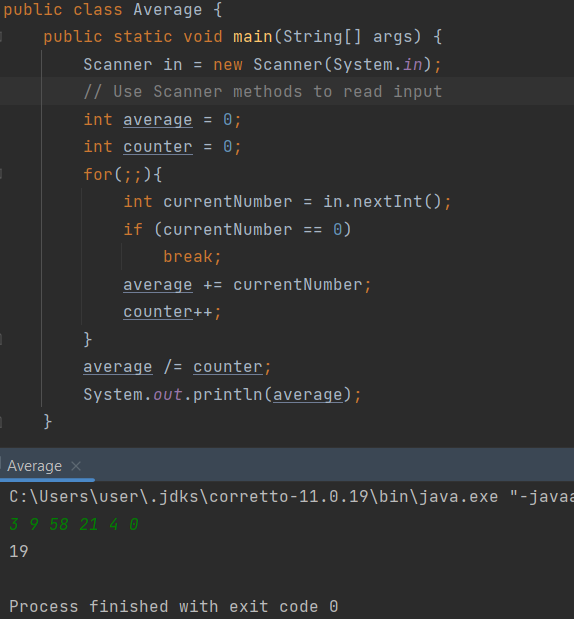
- “Average”. Нужно было написать программу, которая считывает последовательность целочисленных значений из стандартного ввода и находит среднее значение. Пример выполнения программы представлен на рисунке 1.18;

Рисунок 1.18 – Пример выполнения задания “Average”.

- “Pizza Split”. Нужно было написать программу, которая читает два значения – количество людей и количество кусочков в одной пицце. На выходе программа должна напечатать минимальное количество пицц, которое надо заказать, чтобы у всех было одинаковое количество кусочков и не осталось ни

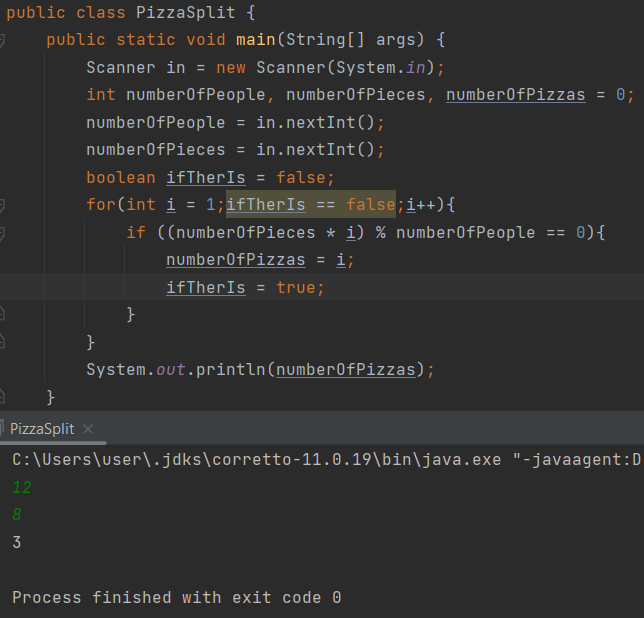
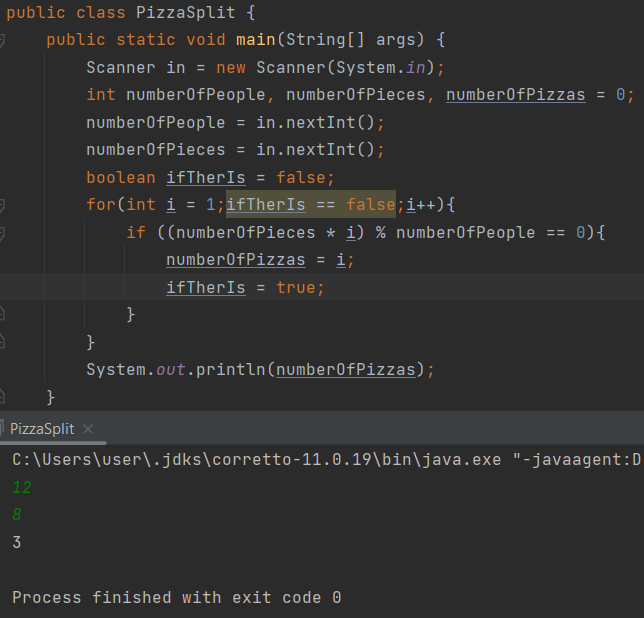
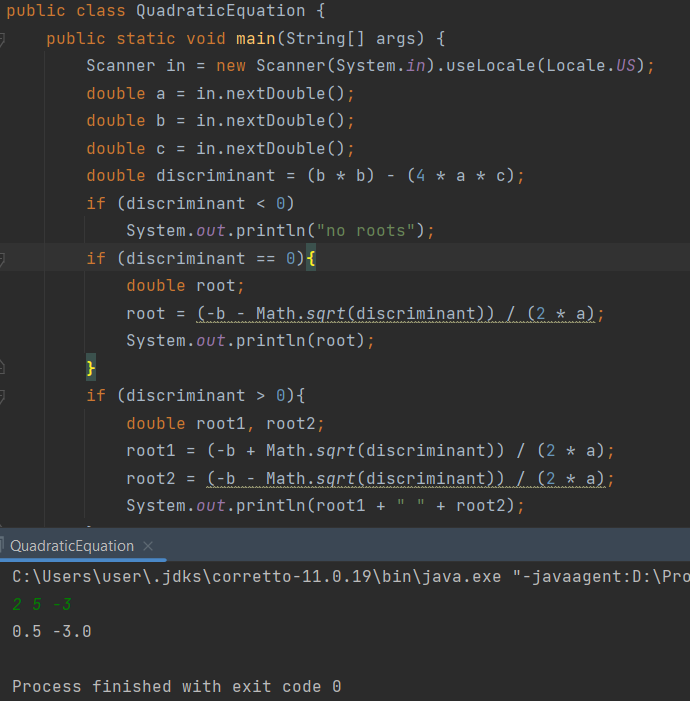
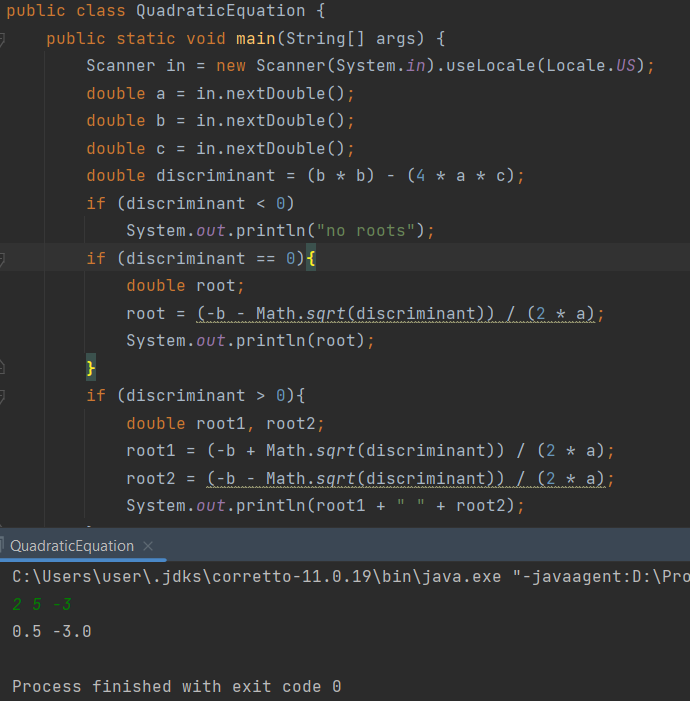
одного лишнего. Пример программы представлен на рисунке 1.19;

Рисунок 1.19 – Пример выполнения программы “Pizza Split”.

- “Quadratic Equation”. Целью задания было научиться разрабатывать программы, требующие ветвления разработки. Нужно было написать программу для решения квадратных уравнений. На вход подаются коэффициенты квадратного уравнения (ax² + bx + c = 0), на выходе печатаются корни уравнения, если они есть. Пример выполнения программы представлен на рисунке 1.20;

Рисунок 1.20 – Пример выполнения программы “Quadratic equation”.

Был изучен четвертый раздел – «Массивы». В теоретической части раздела были рассмотрены следующие подразделы:

- что такое массив. Я изучил концепцию массивов Java и узнал, что массив представляет собой упорядоченную коллекцию элементов одного типа, которые хранятся в памяти последовательно. Понял, как объявлять и инициализировать массивы различных типов данных;

- длина и копирование массивов. Разобрался с понятием длины массива и как получить доступ к его элементам. Изучил методы для копирования массивов, включая использование оператора присваивания и метода ''System.arraycopy()'';

- операторы for и for-each. Работа с элементами массива. Изучил использование оператора for для обхода элементов массива с помощью индексов. Также познакомился с оператором for-each, который позволяет проходить по всем элементам массива без явного использования индексов;

- двумерные массивы. Изучил двумерные массивы, которые представляют собой массивы массивов. Изучил, как создавать, инициализировать и обращаться к элементам двумерных массивов. Узнал о преимуществах и возможностях использования двумерных массивов в различных ситуациях;

- методы класса Arrays. Познакомился с некоторыми полезными методами класса “Arrays”, которые облегчают работу с массивами в Java. Некоторые из них включают сортировку массива, поиск элементов, заполнение массива значениями и другие операции.

В практической части были выполнены предложенные задания:

- “Max method”. Целью задания было научиться работать с массивами. Программа должна была получить массив значений типа int и вернуть максимальный элемент. Реализация метода представлена на рисунке 1.21;

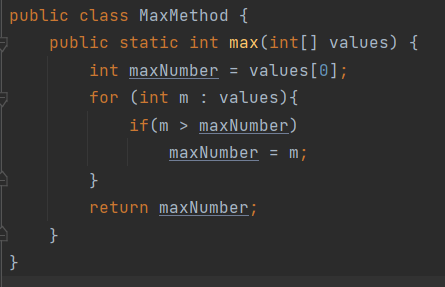


Рисунок 1.21 – Реализация метода “MaxMethod”.

- “Sum of even numbers”. Нужно было реализовать метод, который возвращает сумму четных чисел массива. Реализация метода представлена на рисунке 1.22;

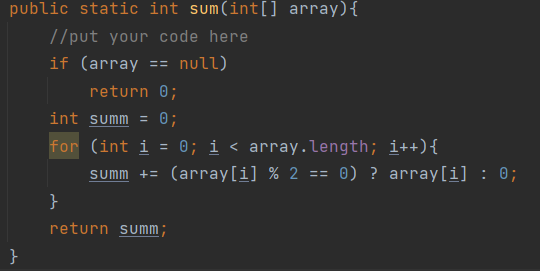


Рисунок 1.22 – Реализация метода “Sum”.

- “Sum of previous”. Нужно было реализовать метод “getSumCheckArray”, который возвращает массив логических значений, где каждый элемент является результатом проверки является ли соответствующий элемент суммой двух предыдущих элементов в данном массиве. Реализация метода представлена на рисунке 1.23;

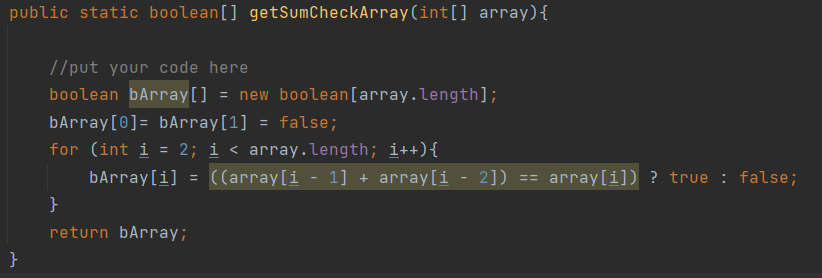


Рисунок 1.23 – Реализация метода “getSumCheckArray”.

- “Cycle Swap”. Нужно было реализовать два статических метода:

- void cycleSwap(int[] array).Сдвигает все элементы в данном массиве вправо на 1 позицию. В этом случае последний элемент массива становится первым;  
- void cycleSwap(int[] array, int shift).Сдвигает все элементы в заданном массиве вправо на shift позиций.

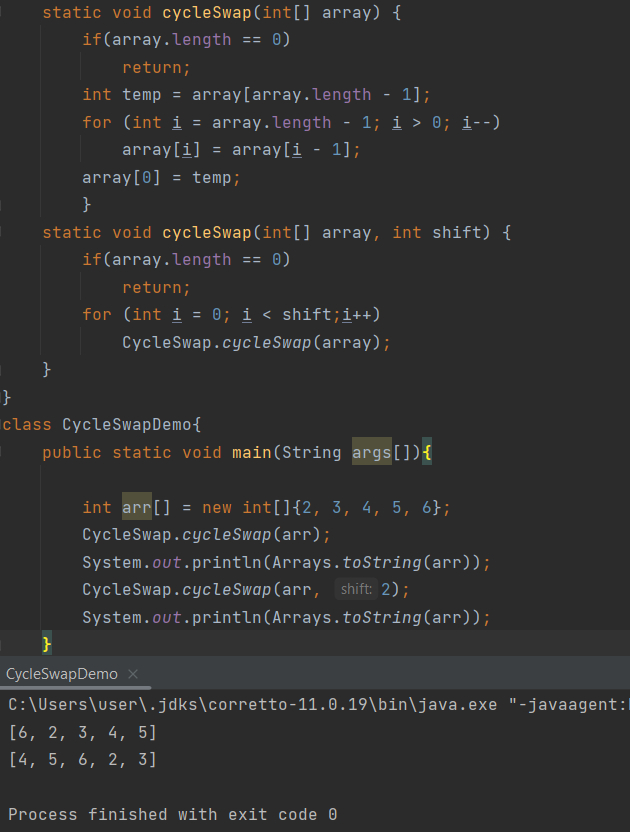
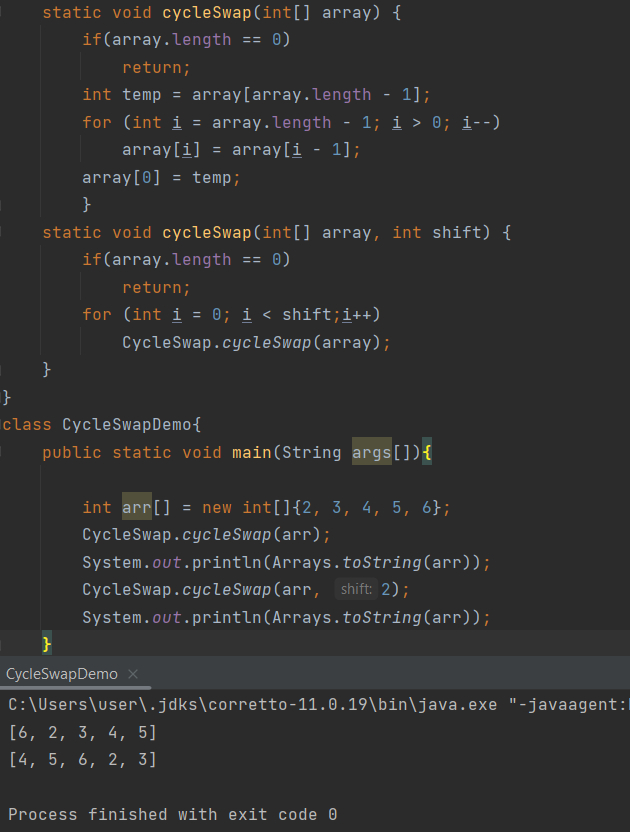
 Реализация методов и пример выполнения программы представлены на рисунке 1.24;

Рисунок 1.24 – Пример выполнения программы “Cycle Swap”.

- “Spiral”. Нужно было реализовать статический метод:

int[][] spiral(int rows, int columns)- возвращает двумерный массив в виде таблицы, содержащий числа от 1 до rows \* columns.   
Числа заполняют "таблицу" по часовой стрелке от верхнего угла по спирали.  
Например, для значений параметров (3, 4), выходной массив должен быть:

1 2 3 4

10 11 12 5

9 8 7 6

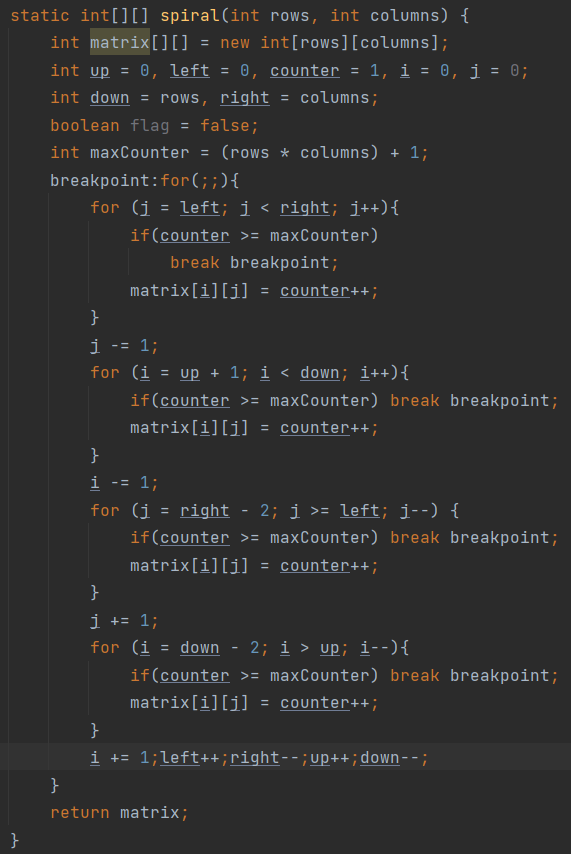
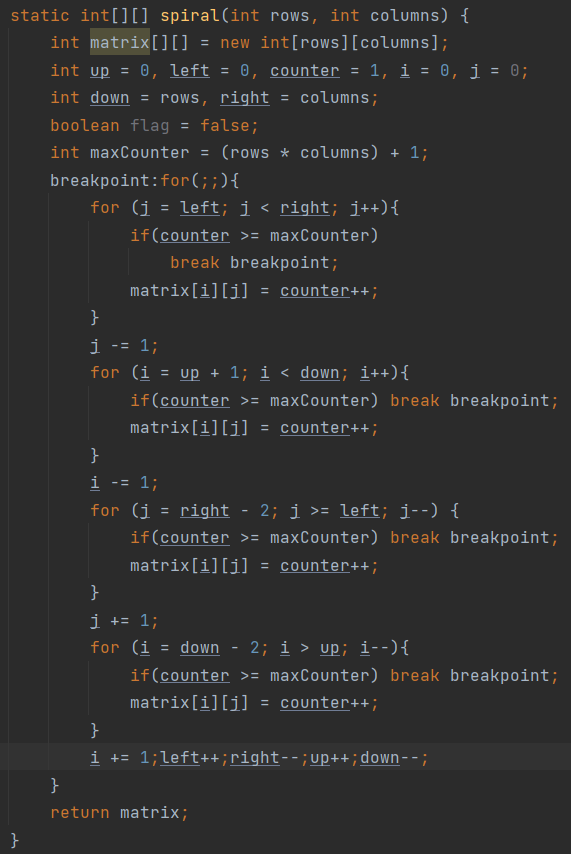
Реализация метода представлена на рисунке 1.25, пример выполнения программы представлен на рисунке 1.26;

Рисунок 1.25 – Реализация метода “spiral”.

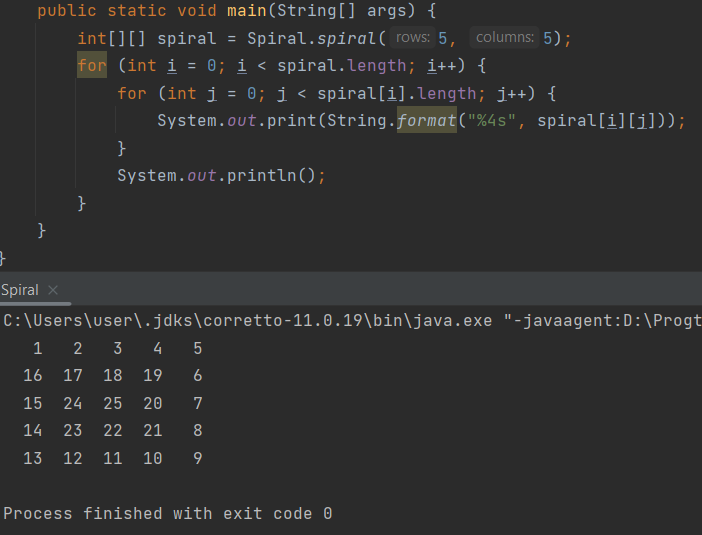


Рисунок 1.26 – Пример выполнения программы “Spiral”.

- “Matrix Transposition”. Целью задания было научиться работать с массивами. Нужно было реализовать метод “multiply”, который в качестве параметра принимает прямоугольный целочисленный массив (матрицу) и возвращает его транспонированным. Реализация метода представлена на рисунке 1.27.

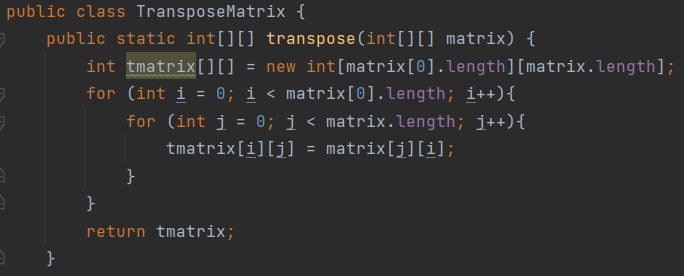


Рисунок 1.27 – Реализация метода “TransposeMatrix”.

- “Matrices Multiplication”. Нужно было реализовать метод “multiply”, который берет перемножает две матрицы. Реализация метода и пример выполнения программы представлены на рисунке 1.28;

Рисунок 1.28 – Пример выполнения программы “Matrices Multiptication”.

**27.03.2023.** Был изучен пятый раздел – «Классы». В теоретической части раздела были рассмотрены следующие подразделы:

- введение в ООП. Получил представление о парадигме объектно-ориентированного программирования и ее основных принципах, таких как инкапсуляция, наследование и полиморфизм;

- классы и объекты. Узнал, что класс – это шаблон или определение, по которому создаются объекты. Понял различие между классами и объектами и как классы используются для создания конкретных экземпляров классов – объектов;

- поля. Изучил понятие поле класса, которые представляют переменные, хранящие состояние объекта. Узнал, как объявлять и использовать поля класса для представления данных объекта;

- методы. Познакомился с концепцией методов класса, которые представляю действия или поведение объекта. Разобрался с объявлением и использованием методов, а также передачей параметров в методы;

- ключевое слово this. Изучил новое ключевое слово “this”, которое используется для обращения к текущему объекту. Узнал, как использовать ''this'' для разрешения конфликтов идентификаторов и доступа к полям и методам объекта;

- создание объектов. Понял, как создавать объекты класса с использованием оператора “new”. Изучил процесс создания объекта, выделение памяти и вызов конструктора;

- конструкторы. Разобрался с понятием конструкторов, которые являются специальными методами класса, вызываемыми при создании объекта. Узнал о различных типах конструкторов и их использовании для инициализации объекта;

- блоки инициализации. Изучил блоки инициализации, которые представляют собой код, выполняемый при создании объекта. Понял, как использовать блоки инициализации для предварительной настройки объектов или классов;

- пакеты. Познакомился с понятием пакетов, которые представляют организационную структуру классов в Java. Узнал о преимуществах использования пакетов для управления и структурирования больших проектов.

изучение классов в Java позволило мне понять основы ООП и использовать классы для организации данных и поведения в программе. Я научился объявлять поля и методы класса, создавать объекты, работать с конструкторами и блоками инициализации. Понятие пакетов дало мне возможность организовывать классы в структурированные группы, что способствует более удобной и эффективной разработке программ на Java.

В практической части были выполнены предложенные задания:

- “Line Intersection”. Целью этого задания было научиться работать с классами и методами. В этом задании было представлено два класса – “Point” и “Line”. Нужно было реализовать метод intersection класса “Line”. Этот метод должен был возвращать точку пересечения двух линий, если она есть.

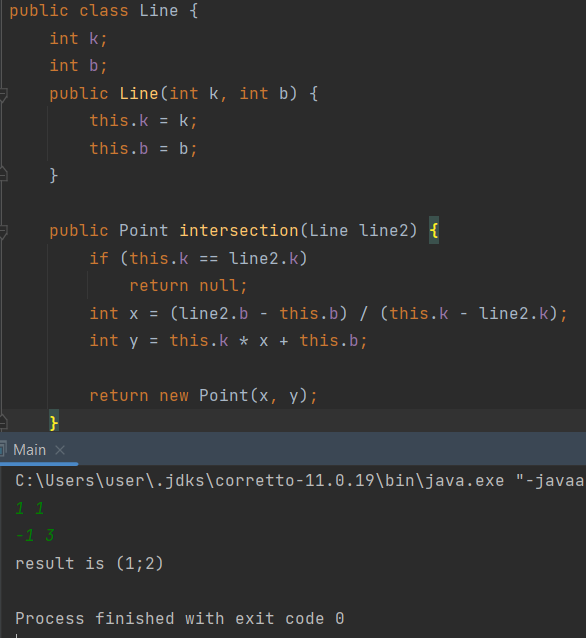
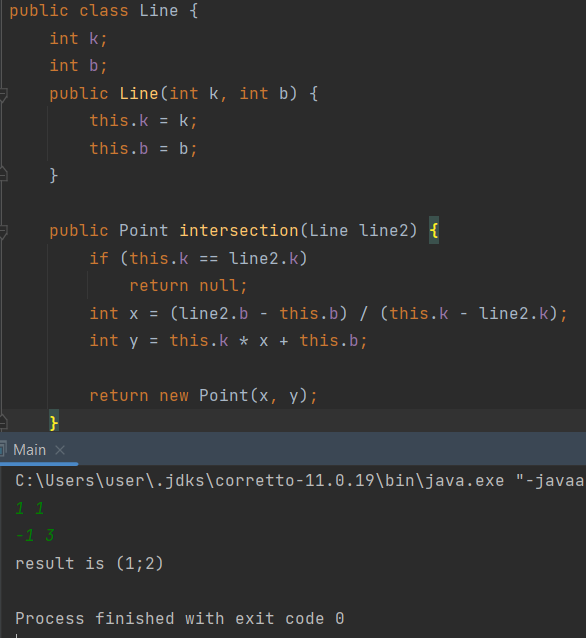
Реализация метода и пример выполнения задания представлены на рисунке 1.29;

Рисунок 1.29 – Пример выполнения программы “Line Intersection”.

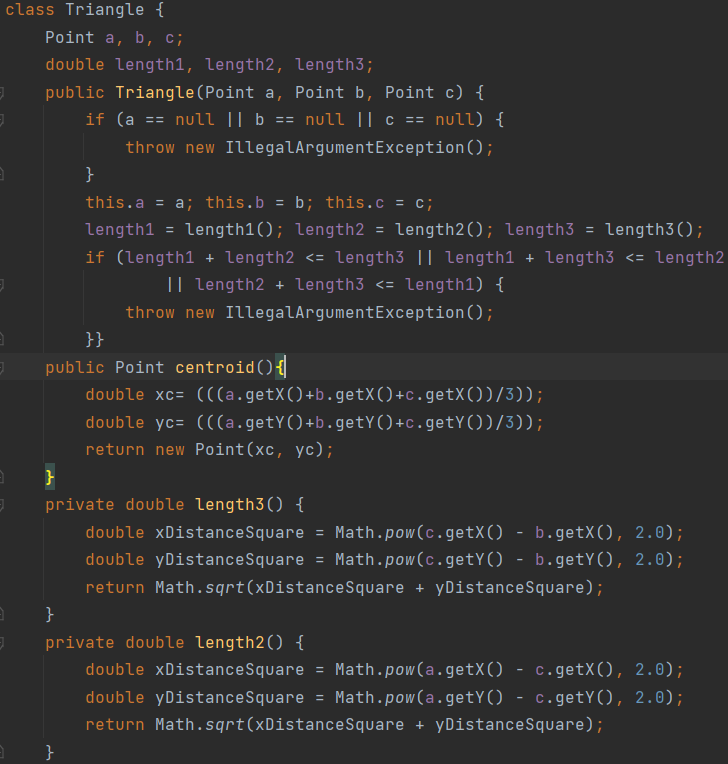
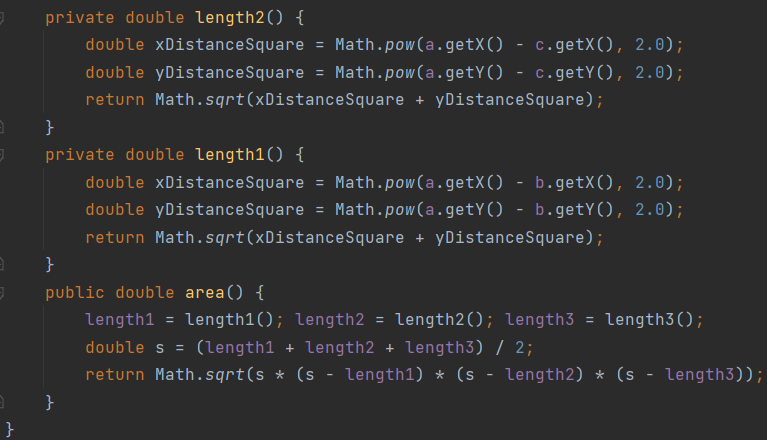
- “Triangle”. В этом задании нужно было реализовать методы класса “Triangle”:

- Конструктор, имеющий в качестве параметров координаты трех вершин;

- double area() – возвращает площадь треугольника;

- Point centroid() – возвращает центроид треугольника.

Реализация класса “Triangle” представлена на рисунке 1.30;

Рисунок 1.30 – Реализация класса “Triangle”.

- “Decrementing Carousel”. Целью этого задания было научиться проектировать простые классы и отношения между ними. Нужно было реализовать два класса – “DecrementingCarousel” и “CarouselRun”. Описание задания: DecrementingCarousel - контейнер, принимающий элементы int. DecrementingCarousel имеет максимальную вместимость, указанную в конструкторе. При создании DecrementingCarousel находится в состоянии накопления: вы можете добавлять элементы с помощью метода addElement и создать объект CarouselRun с помощью метода run. После вызова метода run DecrementingCarousel находится в рабочем состоянии: он отказывается добавлять дополнительные элементы.

CarouselRun позволяет перебирать элементы карусели, уменьшая их один за другим с помощью метода next.

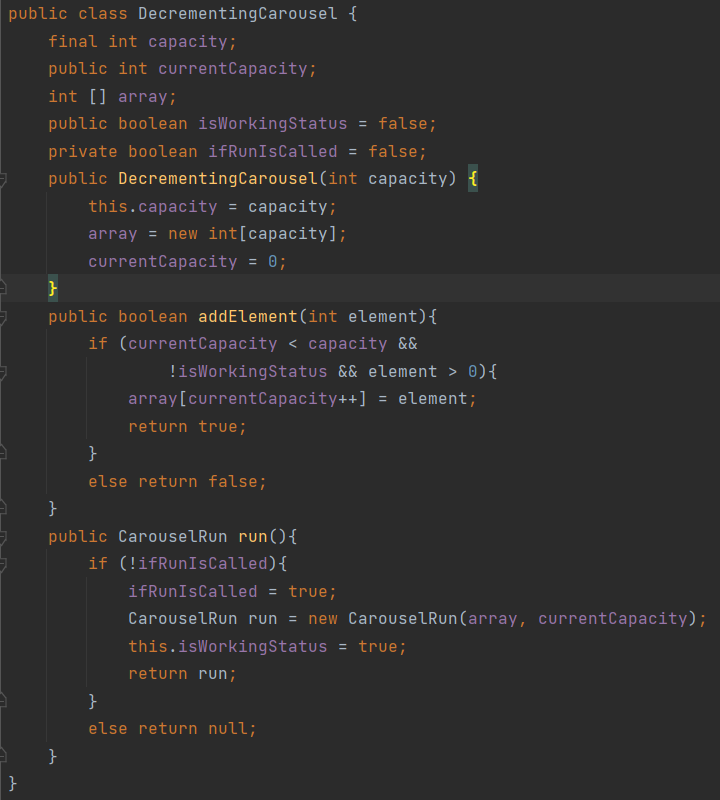
 Реализация классов “DecrementingCarousel” и “CarouselRun” представлена на рисунке 1.31.

Рисунок 1.31 – Реализация классов “DecrementingCarousel” и “CarouselRun”.

Был изучен шестой раздел – «Основы ООП в Java». В теоретической части раздела были рассмотрены следующие подразделы:

- инкапсуляция. Узнал о концепции инкапсуляции, которая позволяет скрыть внутреннюю реализацию объекта и обеспечить доступ к нему только через определенные методы. Понял, как объявлять приватные поля класса, а также использовать геттеры и сеттеры для доступа к этим полям;

- модификаторы static и final. Изучил модификаторы, которые позволяют определить статические поля и методы класса, а также неизменяемые значения;

- наследование. Часть 1. Изучил понятие наследования, которое позволяет создавать новые классы на основе существующих.

- наследование. Часть 2. Разобрался с дополнительными аспектами наследования, включая модификаторы доступа, переопределение методов, вызов конструктора суперкласса;

- полиморфизм. Изучил понятие полиморфизма, которое позволяет объектам разных классов проявлять различное поведение при вызове одного и того же метода;

- перегрузка методов. Разобрался с понятием перегрузки методов, которая позволяет иметь несколько методов с одним и тем же именем, но с различными параметрами;

- класс Object. Узнал о классе Object, который является базовым для всех классов в Java;

Изучение этих основ ООП в Java позволило мне понят принципы инкапсуляции, наследования и полиморфизма, которые являются ключевыми для разработки гибких и масштабируемых приложений на Java.

В практической части были выполнены предложенные задания:

- “Halving Carousel”. В этом задании необходимо было расширить класс “DecrementingCarousel” из прошлого задания. Этот подкласс должен был делить элементы пополам, а не уменьшать их на единицу. Реализация класса представлена на рисунке 1.32;

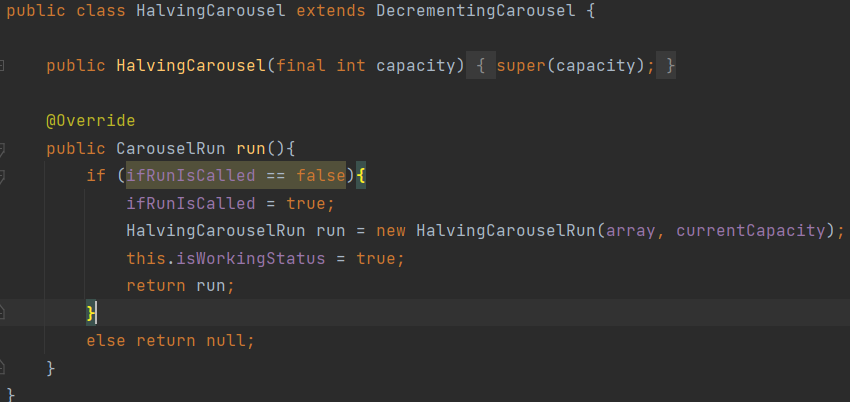


Рисунок 1.32 – Реализация класса ''HalvingCarousel''.

- “Gradually Decreasing Carousel”. В этом задании нужно было расширить класс “DecreasingCarousel”. Новый подкласс должен был уменьшать элементы путем постепенного увеличения уменьшения. Реализация класса представлена на рисунке 1.33.

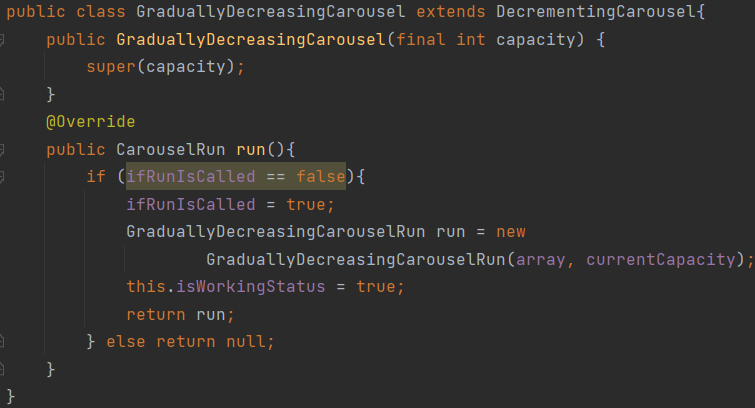


Рисунок 1.33 – Реализация класса “GraduallyDecreasingCarousel”.

**03.04.2023.** Был изучен седьмой раздел – «Абстрактные классы и интерфейсы». В теоретической части разделы были рассмотрены следующие подразделы:

- абстрактные классы. Узнал о понятии абстрактных классов, которые представляют собой классы, от которых нельзя создавать объекты на прямую. Абстрактные классы могут содержать как абстрактные методы, т.е. методы без реализации, так и конкретные методы;

- интерфейсы. Изучил концепцию интерфейсов, которые представляют собой набор абстрактных методов, которые классы должны реализовать. Понял, как объявлять интерфейсы с помощью ключевого слова “interface” и как классы реализуют интерфейсы с помощью ключевого слова “implements”;

- интерфейсы в Java 8. Узнал о новых возможностях интерфейсов, введенных в Java 8. Изучил использование дефолтных методов, которые представляют реализацию по умолчанию для методов интерфейса;

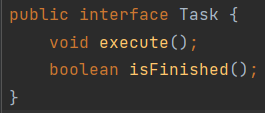
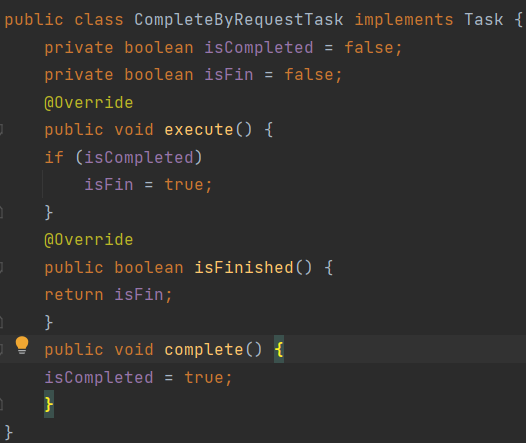
- клонирование объектов. Разобрался с понятием клонирования объектов, которое позволяет создавать копии существующих объектов. Изучил использование интерфейса “Cloneable” и метода clone() для реализации клонирования объектов в Java;

- рекомендации при проектировании. Получи рекомендации и советы при проектировании классов, абстрактных классов и интерфейсов и других принципах проектирования, которые позволяют создавать гибкий, расширяемый и поддерживаемый код.

Изучение абстрактных классов и интерфейсов в Java позволило мне понять принципы абстракции и разделения ответственности, а также использования их для создания гибких и масштабируемы программных решений.

В практической части раздела были выполнены предложенные задания:

- “Task Carousel”. Целью задания было научиться проектировать реализации интерфейса и использовать из. В задании нужно было спроектировать две реализации интерфейса “Task”. Реализация задания представлена на рисунке 1.34.

Рисунок 1.34 – Интерфейс “Tast” и его реализации.

Был изучен восьмой раздел – «Вложенные классы». В теоретической части раздела были рассмотрены следующие подразделы:

- понятие вложенного класса. Узнал о вложенных класса, которые представляют собой классы, объявленные внутри других классов. Вложенные классы могут быть использованы для логической группировки классов;

- внутренние классы. Изучил внутренние классы, которые являются нестатическими классами, объявленными внутри другого класса. Внутренние классы имеют доступ к полям и метода внешнего класса и могут быть использованы для реализации связи между классами;

- локальные классы. Разобрался с понятием локальных классов, которые объявляются внутри блока кода, такого как метод. Локальные классы имеют доступ к переменным внешнего блока кода и могут быть полезны для решения конкретных задач внутри методов;

- анонимные классы. Узнал об анонимных классах, которые представляют собой безымянные классы, создаваемы на месте их использования. Анонимные классы обычно используются для реализации интерфейсов или расширения абстрактных классов без явного создания нового класса;

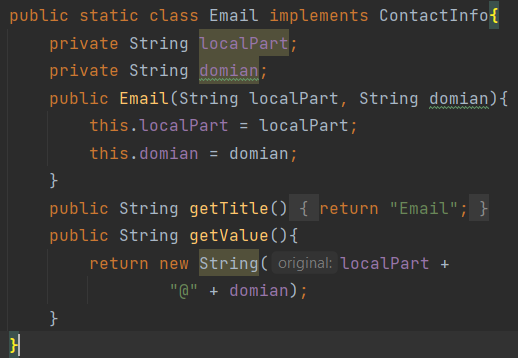
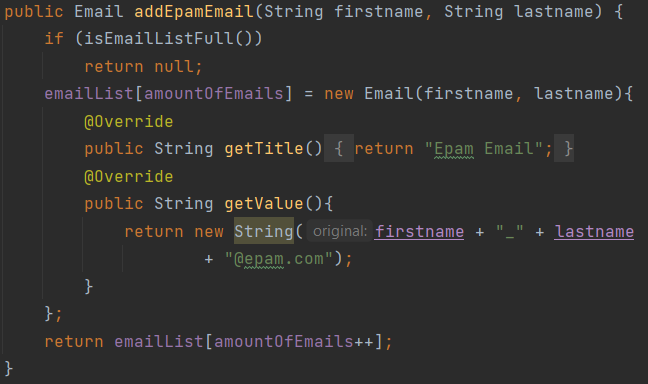
- статические вложенные классы. Изучил статические вложенные классы, которые являются статическими классами, объявленными внутри другого класса. Статические вложенные классы не имеют доступа к нестатическим полям и методам внешнего класса;

- особенности вложенных классов. Разобрался с особенностями вложенных классов, такими как доступность полей и методов внешнего класса, способы создания экземпляров вложенных классов и их взаимодействие с внешним классом.

Изучение вложенных классов в Java позволило мне лучше организовать код, улучшить его читаемость и модульность. Вложенные классы также представляют собой удобный способ объединения связанных классов внутри одного файла и позволяют создавать более гибкие и масштабируемые приложения.

В практической части было выполнено 2 предложенных задания:

- “Contact Book”. Целью задания было научится использовать вложенные классы. В этом задании в классе “Contact”, который содержит контактную информацию о человеке нужно было реализовать внутренние классы, записывающие различные виды контактной информации. Реализация вложенных классов представлена на рисунке 1.35.

Рисунок 1.35 – Реализаци внутренних методов класса “Contact”.

**10.04.2023.** Был изучен девятый раздел – «Работа со строками». В теоретической части раздела были рассмотрены следующие подразделы:

- класс String. Класс String в Java представляет собой неизменяемую последовательность символов. Изучив класс String, я узнал о различных операциях, которые можно выполнять над строками, таких как сравнение строк, получение подстроки, конкатенация строк и другие;

- классы StringBuilder и StringBuffer. Эти классы предоставляю более эффективные способы изменения строк, особенно при выполнении множества операций. Класс StringBuilder предназначен для использования в однопоточных сценариях, в то время как класс StringBuffer предназначен для использования в многопоточных сценариях;

- регулярные выражения. В этом подразделе были изучены основы использования регулярных выражений в Java, включая специальные символы и метасимволы, которые позволяют задавать шаблоны поиска. Узнал, как использовать класс Pattern и класс Matcher для выполнения операций поиска, сопоставления и замены текста на основе регулярных выражений.

В практической части раздела были выполнены предложенные задания:

- “Validation. Color Code”. В этом задании нужно было реализовать метод, который проверяет входную строку на соответствие правилам написания HTML Color Codes. Реализация метода представлена на рисунке 1.36.

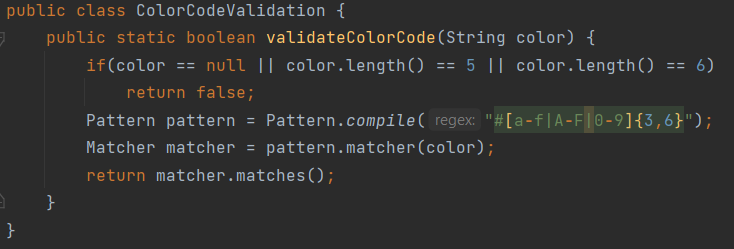


Рисунок 1.36 – Реализация метода “validateColorCode”.

Был изучен десятый раздел – «Исключения». В теоретической части раздела были рассмотрены следующие подразделы:

- исключения и их типы. Узнал, что исключения представляют собой события, которые возникают во время выполнения программы и могут привести к изменению нормального потока выполнения. Изучи различные типы исключений в Java;

- обработка исключений. Узнал о механизмах обработки исключений, которые позволяют программисту контролировать и реагировать на исключительные ситуации;

- выбрасывание и распространение исключений. Рассмотрел способы выбрасывания исключений с помощью ключевого слова throw и как исключения распространяются по стеку вызовов методов, пока не будут перехвачены и обработаны;

- собственные исключения. Узнал о возможности создания собственных исключений в Java. Изучил, как объявить и определить своей класс исключения, наследуемы от класса “Exception” или его подклассов.

В результате изучения этого раздела я получил понимание важности обработки исключений для создания надежных и стабильных программ.

В практической части раздела были выполнены 2 предложенных задания:

- “Requirements”. Целью задания было отточить навыки работы с исключениями и их выбрасыванием. Нужно было реализовать методы класса “Requirements”, которые в зависимости от условий выбрасывают исключения. Реализация методов представлена на рисунке 1.37;

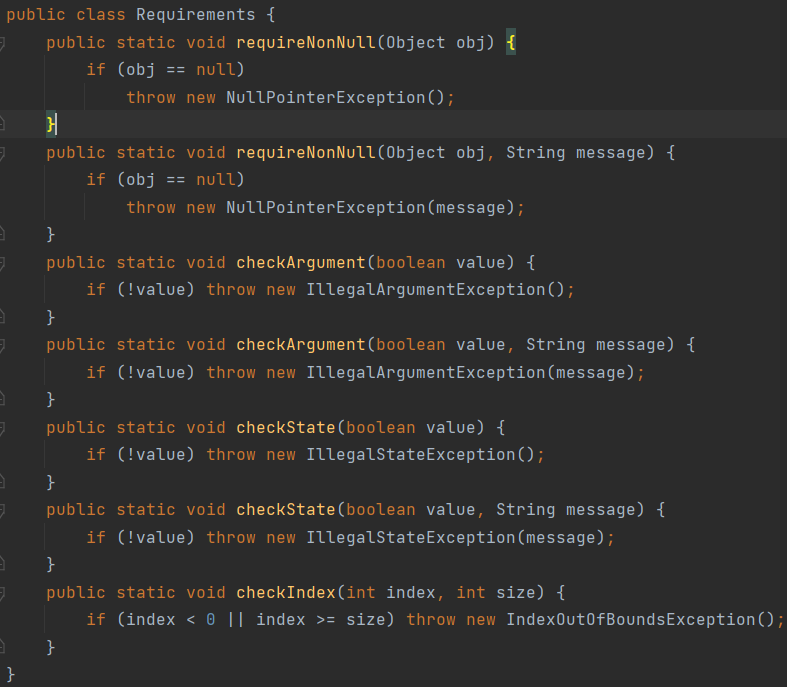


Рисунок 1.37 – Реализация методов класса “Requirements”.

- “Catch’em all”. Целью задания было отточить навыки работы с исключениями и научиться их перехватывать. В этом задании нужно было реализовать метод main в классе CatchEmAll. Это метод должен обрабатывать вызов метода riskyMethod, который может бросать различные типы исключений. Также следовало учитывать, что некоторые типы исключений стоило обрабатывать, а некоторые – нет. Реализация задания представлен на рисунке 1.38.

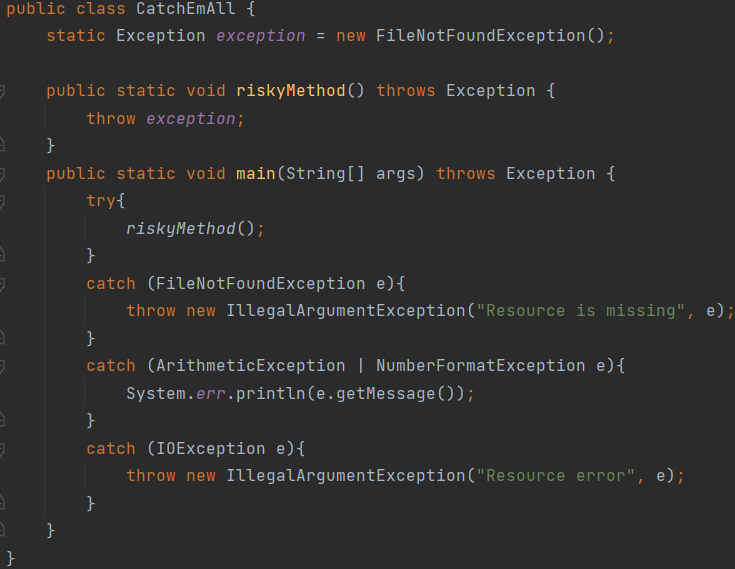


Рисунок 1.38 – Реализация задания “CatchEmAll”.

**17.04.2023.** Был изучен одиннадцатый раздел – «Аннотации». Данный раздел был представлен только для ознакомления, поэтому не содержал практический заданий, а только теоретическую часть. В теоретической части были рассмотрены следующие подразделы:

- аннотации. Узнал, что аннотации являются метаданными, которые можно добавлять к коду для предоставления дополнительной информации и инструкций компилятору или другим инструментам. Познакомился с такими аннотациями как “@Override”, “@Deprecated”, “@SupressWarning”. Также изучил создание пользовательских аннотаций с помощью ключевого слова “@interface”.

Был изучен двенадцатый раздел – «Обобщения». В теоретической части раздела были рассмотрены следующие подразделы:

- обобщение типов. Узнал, что обобщения позволяют создавать классы и методы, которые могут работать с различными типами дынных, обеспечивая безопасность типов во время компиляции. Понял, как объявлять обобщенные классы, а также как использовать их для работы с различными типами данных;

- обобщение методов. Узнал, что обобщение методов позволяет создавать методы, которые могут принимать параметры типа и возвращать значения обобщенного типа. Изучил синтаксис и примеры использования обобщенных методов;

- ограничения и подчистка. Обобщения и наследование. Ограничение типов позволяет ограничить диапазон типов, которые могут быть использованы в обобщенных классах или методах;

- подстановочный тип. Подстановочный тип представляет собой неизвестный тип. Он позволяет работать с обобщенными классами или методами без жестокой привязки к определенному типу.

В практической части раздела было выполнено предложенное задание “Beware of Dogs”. В этом задании нужно было, используя обобщения, изменить класс “House” таким, образом, чтобы собаки не могла входить в кошачий домик, а кошки не могла входить в домик для собак. Реализация Класса “House” представлена на рисунке 1.39.

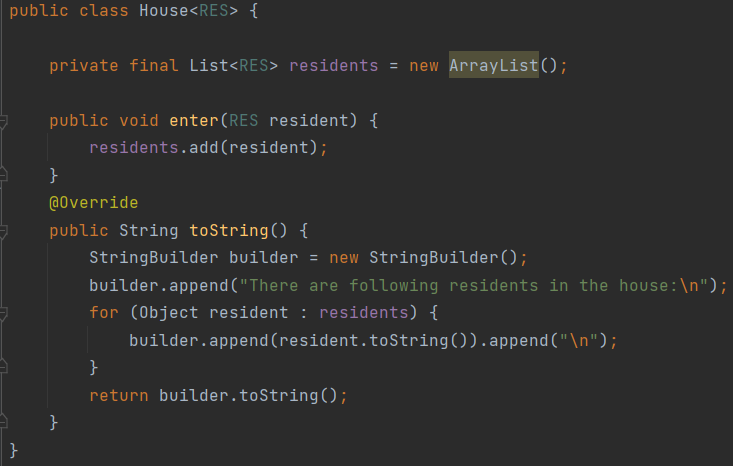


Рисунок 1.39 – Реализация класса “House”.

**24.03.2023.** Был изучен тринадцатый раздел – «Перечисления». В теоретической части раздела были изучены следующие подразделы:

- перечисления. Перечисления в Java представляют собой специальный тип данных, который позволяет определить набор констант с заданными значениями. Они обеспечивают лучшею структуру и понятность кода, позволяя организовать группу связанных констант в одном месте. Перечисления также могут содержать статические методы и константы, которые доступны через имя перечисления.

В практической части раздела было выполнено предложенное задание “Compass”. В этом задании нужно было реализовать методы:

- ofDegrees – возвращает Direction по входным градусам;

- closestToDegrees – возвращает ближайший к входным градусам экземпляр Direction;

- opposite – возвращает противоположный указанному Direction;

- differenceDegreeTo – возвращает разницу в градусах между указанным и входным экземпляром Direction.

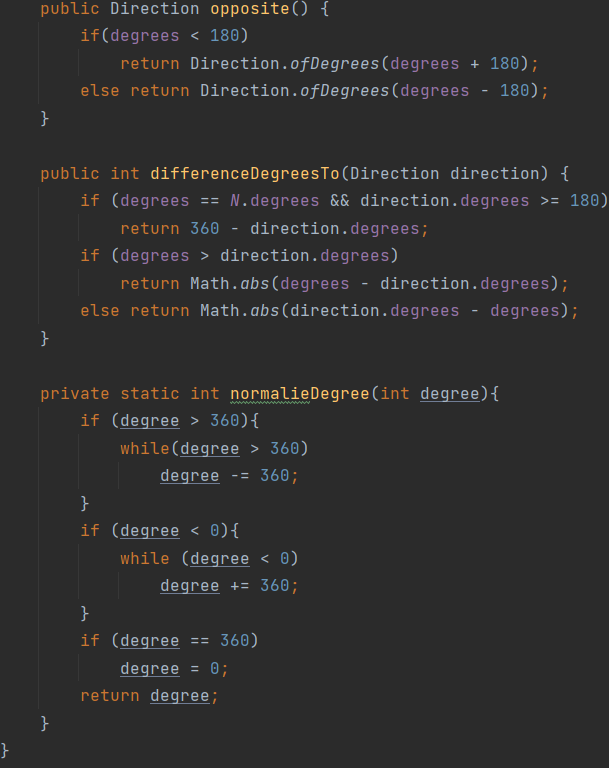
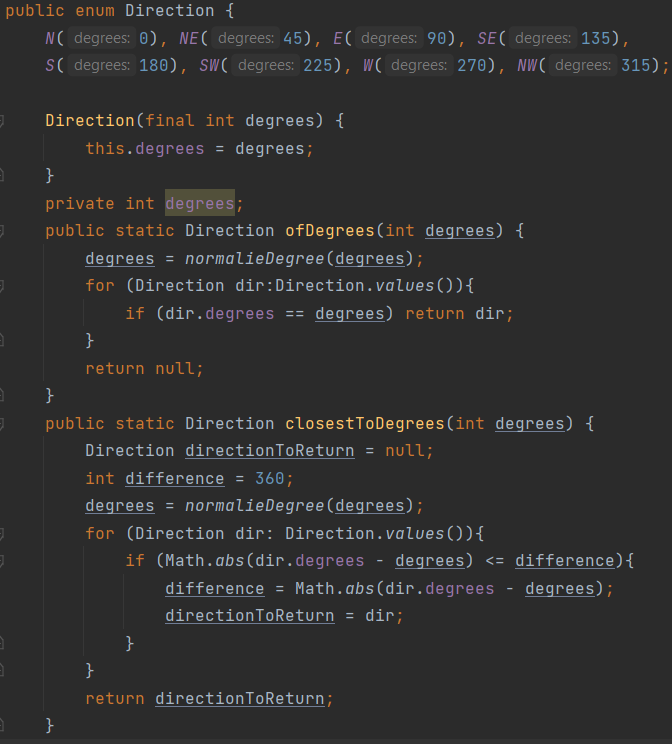
 Реализация класса “Direction” представлена на рисунке 1.40.

Рисунок 1.40 – Реализация класса “Direction”.

**03.05.2023.** Был изучен четырнадцатый раздел – «Классы-оболочки». В теоретической части раздела были рассмотрены следующие подразделы:

- что такое класс-оболочка. Узнал, что класс-оболочка в Java представляет собой класс, который инкапсулирует примитивные типы данных и предоставляет дополнительные методы и функциональность для работы с ними;

- создание объектов классов-оболочек. Научился создавать объекты классов-оболочек, в том числе используя конструкторы и статические методы, предоставленные классами-оболочками;

- механизмы автоупаковки и автораспаковки. Познакомился с механизмами автоупаковки и автораспаковки в Java, которые автоматически преобразуют примитивные типы в соответствующие классы-оболочки и наоборот;

- классы-оболочки для высокоточной арифметики. Узнал, что в Java предоставляются классы-оболочки для работы с высокоточной арифметикой, такие как BigDemical и BigInteger. Эти классы позволяют работать с числами большой точнисти и обеспечивают точное представление и выполнение арифметических операций с большими числами.

В результате изучения этого раздела я понял, насколько полезны классы-оболочки во множестве ситуаций, например, при необходимости работы с примитивными типами данных в контексте объектно-ориентированного программирования. Они обеспечиваю дополнительные возможности и функциональности для работы с данными и облегчают взаимодействие между примитивными типами и объектами.

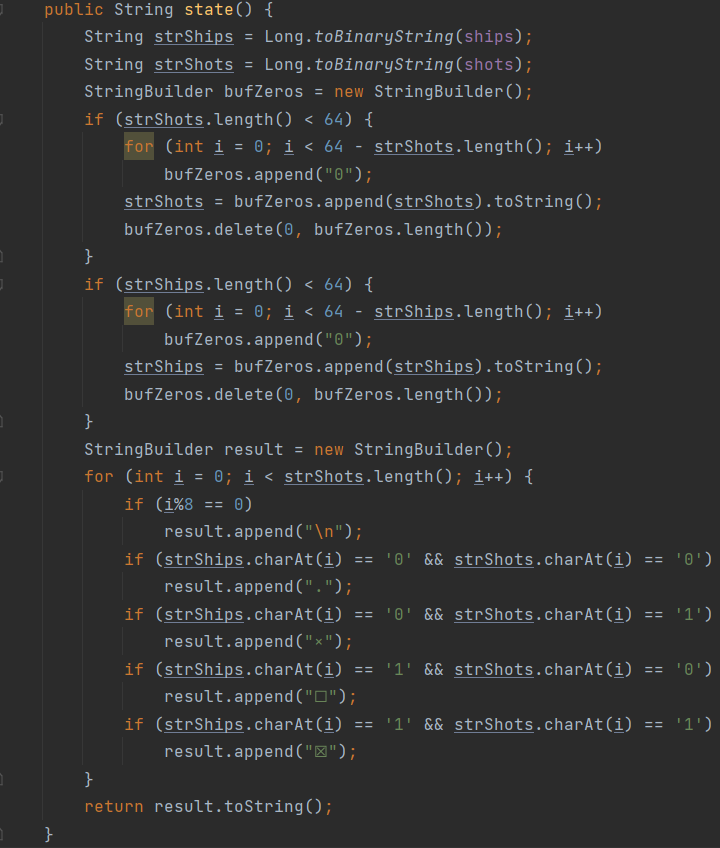
В практической части раздела было выполнено предложенное задание “Battleship8x8”. В этом задании нужно было реализовать методы класса “Battleship8x8” для игры в морской бой. Реализация класса представлена на рисунке 1.41.

Рисунок 1.41 – Реализация класса “Battleship8x8”.

**15.05.2023.** Был изучен пятнадцатый раздел – «Класс Optional». В теоретической части раздела были рассмотрены следующие разделы:

- класс Optional. Этот класс предоставляет удобный способ работы с возможно отсутствующими значениями. Он представляет собой контейнер, который может содержать либо ненулевое значение, либо быть пустым. Это полезный инструмент для обработки ситуаций, когда значение может отсутствовать, и позволяет избежать использования значений null;

- методы класса Optional. Познакомился с некоторыми методами, которые предоставляет класс Optional, которые позволяют более безопасно и удобно работать с возможно отсутствующими значениями, избегая ошибок, связанных с нулевыми значениями.

В практической части раздела было выполнено предложенное задание “Optional Max”. Целью этого задания было научится работать с классом Optional. В задании нужно было реализовать метод MaxMethod, который получает массив целочисленных значений и возвращает его максимальное значение в виде OptionalInt. Реализация метода представлена на рисунке 1.42.

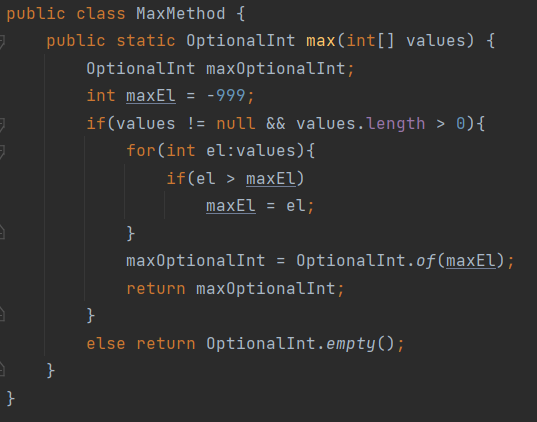


Рисунок 1.42 – Реализация задания “MaxMethod”.

Был рассмотрен шестнадцатый раздел – «Документирование кода». Данный раздел был представлен только для ознакомления, поэтому не содержал практических заданий, а только теоретическую часть. В теоретической части раздела были рассмотрены следующие подразделы:  
 - комментарии и документирование. Рассмотрел различные виды комментариев, такие как однострочные и многострочные комментарии. Особое внимание было уделено документированию кода с использованием комментариев, соответствующих определенным соглашениям и стандартам, что позволяет автоматически генерировать документацию на основе комментариев и предоставляет полезную информацию о классах, методах, параметрах и возвращаемых значениях;

- генерирование документации. Ознакомился с инструментом Javadoc для автоматической генерации документации на основе комментариев в исходном коде. Узнал, что после написания комменариев с ипользованием правлиных тегов и форматирования, Javadoc может быть запущен, чтобы сгенерировать HTML-документацию, которая может быть легко просмотрена и доступна для других разработчиков. Понял, что генерирование документации является хорошей практики разработки, поскольку облегчает понимание и использование кода другими программистами, а также повышает поддерживаемость программного кода и улучшает совместную работы над проектом.

**22.05.2023.** Был пройден финальный тест по курсу, который включал 60 вопросов. Результат представлен на рисунке 1.43.

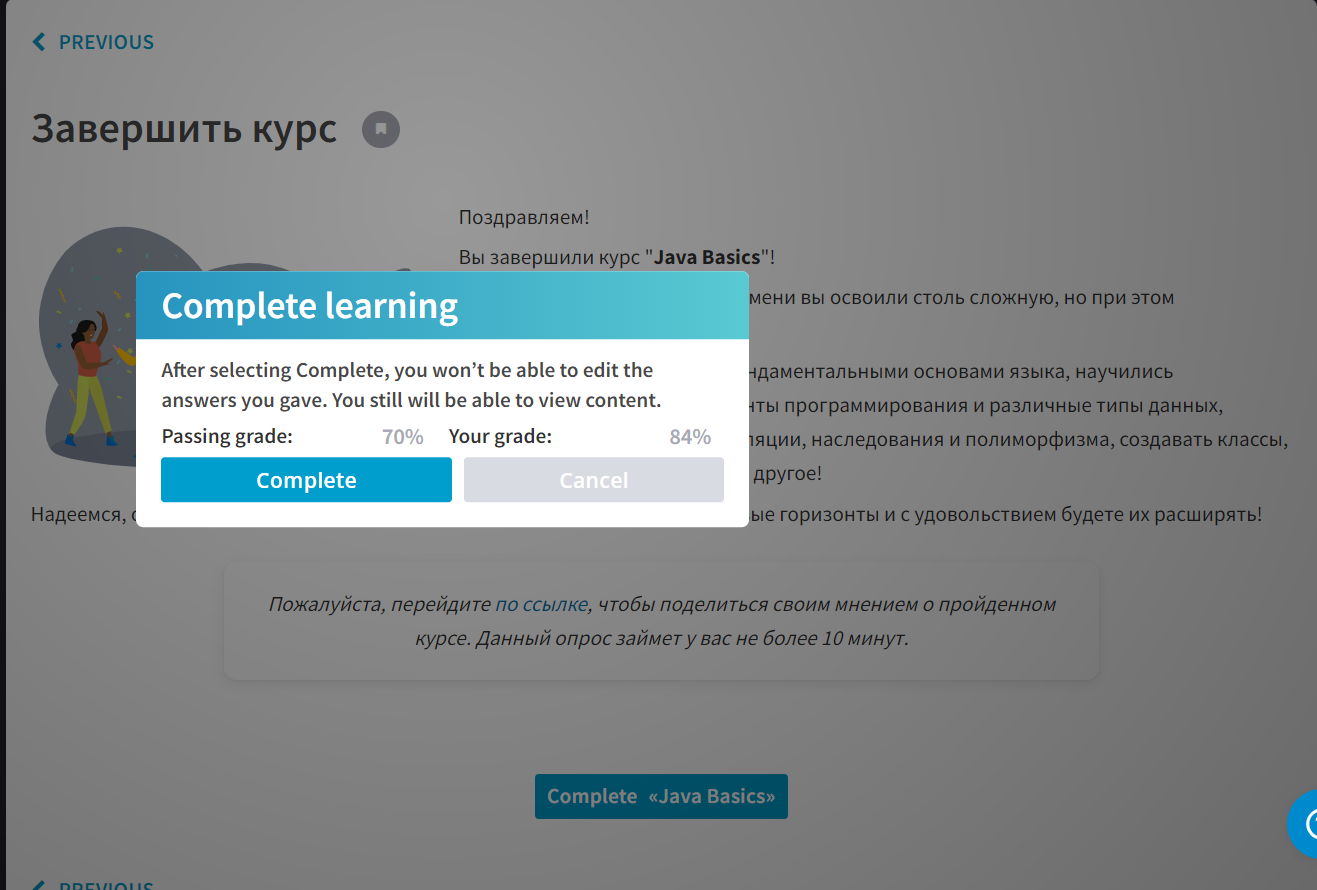


Рисунок 1.43 – результат выполнения финального теста.

После выполнения финального теста тренинг был успешно выполнен и завершен.

**Вывод:**В ходе ознакомительной учебной практики были изучены основные концепции и элементы языка программирования Java. Курс включал изучение различных разделов, включая «Типы данных», «Условия и циклы», «Классы», «Основы ООП», и другие.

В каждом разделе были рассмотрены основные понятия, а также проведена практическая работа, включающая выполнение заданий. В результате были закреплены основные концепции программирования на языке Java. Кроме того, в ходе учебной практики был пройден финальный тест, результаты которого подтвердили уровень полученных знаний и навыков.

В целом, ознакомительная учебная практика была полезным и продуктивным опытом. Теоретический материал и практические задания позволили получить фундаментальные знания о языке программирования Java. Полученные знания и навыки являются хорошей отправной точкой для дальнейшего изучения Java и разработки программных решений на этом языке.