|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет»  РТУ МИРЭА |  |

Институт информационных технологий

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

Практические РАБОТы

по дисциплине «Программирование на языка Джава»

Выполнил студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шило Ю.С.

(ИКБО-33-22)

Принял старший преподаватель Рачков А.В.

Практические работы работа выполнены «\_\_\_»\_\_\_\_\_\_\_2023г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2023г.

Москва 2023

СОДЕРЖАНИЕ

[1 Практическая работа №1 3](#_Toc145667124)

[2 Практическая работа №2 11](#_Toc145667125)

[2 Практическая работа №2 11](#_Toc145667126)

[3 Практическая работа №3 20](#_Toc145667127)

[4 Практическая работа №4 25](#_Toc145667128)

# 1 Практическая работа №1

**1.1 Цель работы**

Введение в разработку на языке программирования Java. Скачать и установить среду разработки IntelliJ IDEA, разобраться в базовом синтаксисе и конструкциях, существующих в языке Java и выполнить предложенный список работ, создать свой Git репозиторий и залить на него выполненные работы.

**1.2 Теоретические сведения**

Язык Джава— это объектно-ориентированный язык программирования, с инкапсуляцией и со строгой типизацией. Программы, написанные на языке, Джава могут выполняться под управлением различных операционных системах при наличии необходимого ПО – Java Runtime Environment.

Для того чтобы создать и запускать программы на языке Джава необходимо следующее ПО:

1. Java Development Kit (JDK);
2. Java Runtime Environment (JRE);
3. cреда разработки (в нашем случае будет использоваться IntelliJ IDEA).

**1.3 Выполнение лабораторной работы**

*Задание 1:*

Создать проект в IntelliJ IDEA.

*Решение:*

Для решения задания нам потребуется скачать IntelliJ IDEA. Для того чтобы создать проект в IntelliJ IDEA нам нужно зайти (рисунок 1.1) в раздел “File”, затем в раздел “New” и нажать на “Project”

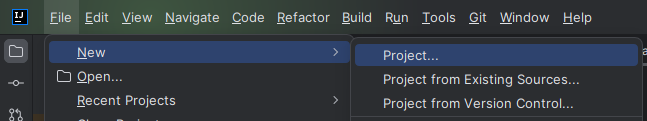


Рисунок 1.1- Раздел для создания нового проекта в IntelliJ IDEA

В открывшимся окне (рисунок 1.2), мы можем выбрать название в разделе “Name” и путь, по которому у нас будет располагаться проект в разделе “Location”.

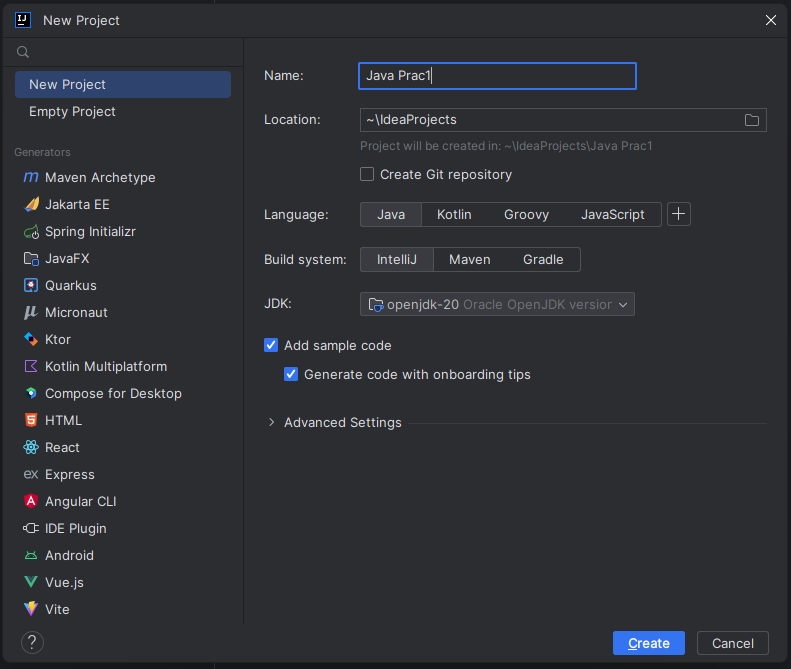


Рисунок 1.2 – Окно создания нового проекта в IntelliJ IDEA

Написав нужное название и путь для проекта нажимаем кнопку “Create”. Открывшиеся окно (рисунок 1.3) является новый проектом.

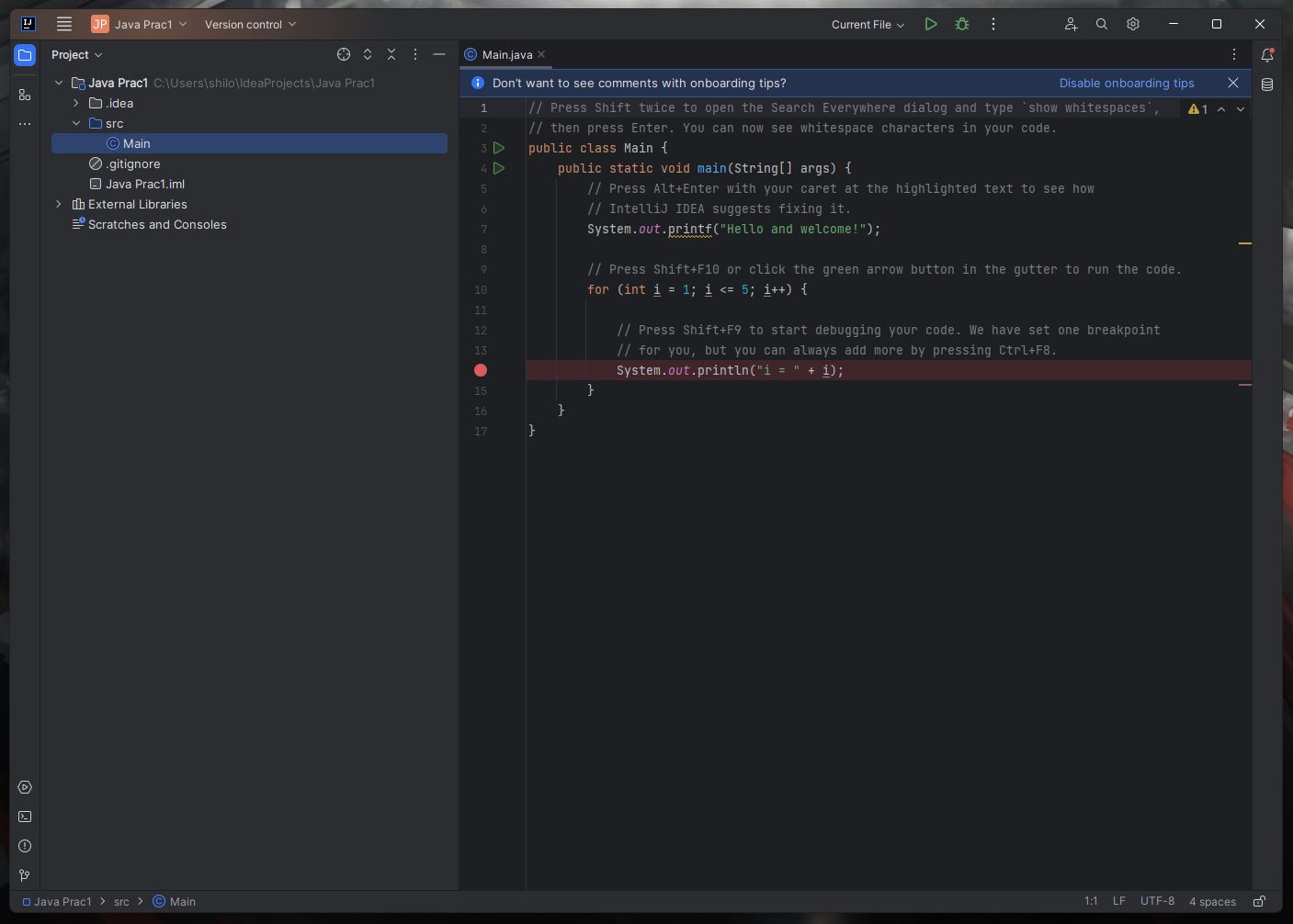


Рисунок 1.3 – Окно, созданного проекта в IntelliJ IDEA

*Задание 2:*

Создать свой собственный Git репозитарий.

*Решение:*

Для решения данной задачи нам необходимо зайти или создать аккаунт на сайте GitHub (URL: <https://github.com/>). После этого необходимо нажать кнопку “New” (рисунок 1.4).

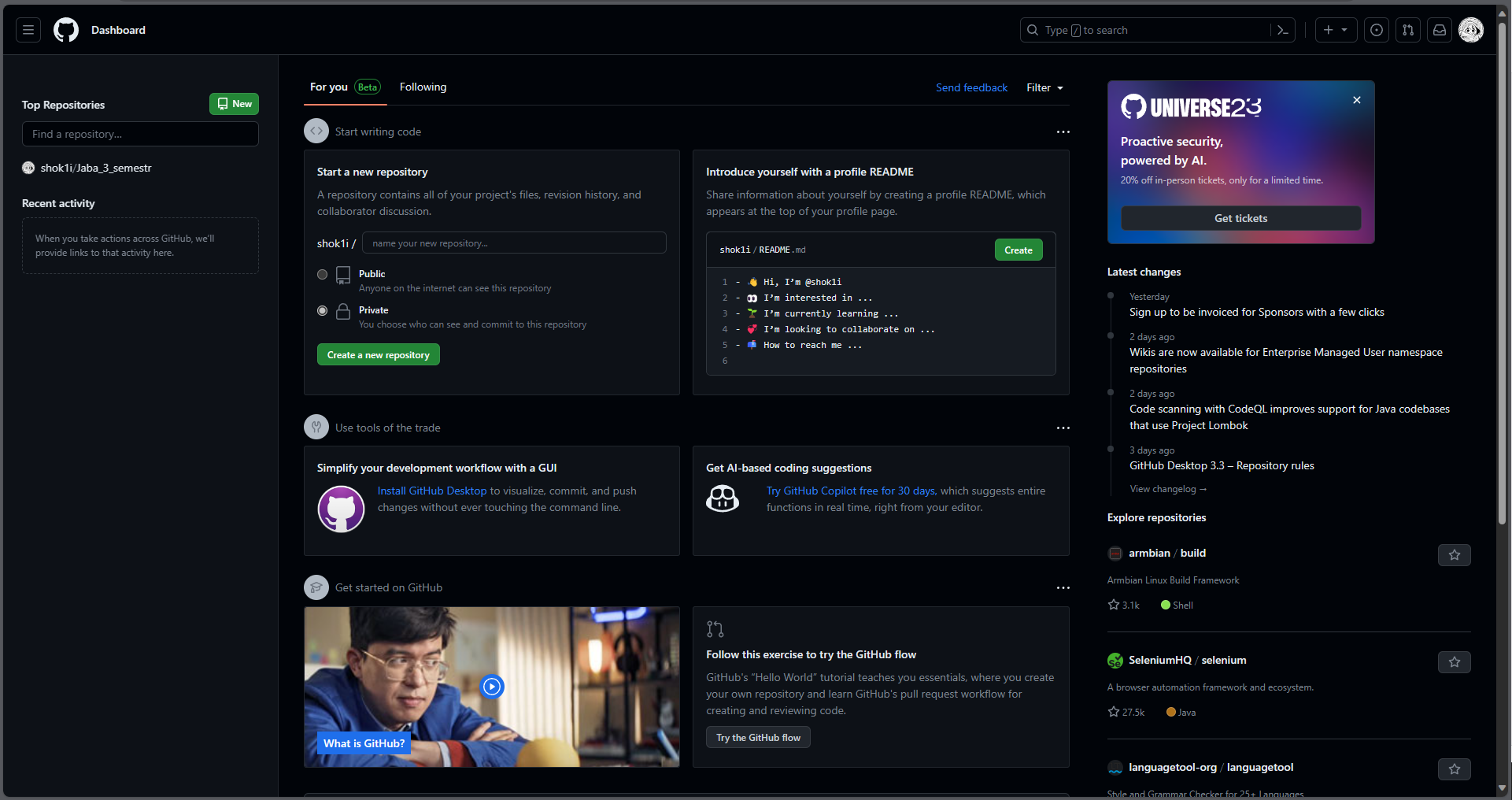


Рисунок 1.4 – Главная страница сайта GitHub

Далее требуется ввести название в разделе “Repository name” и нажать “Create repository” (Рисунок 1.5).

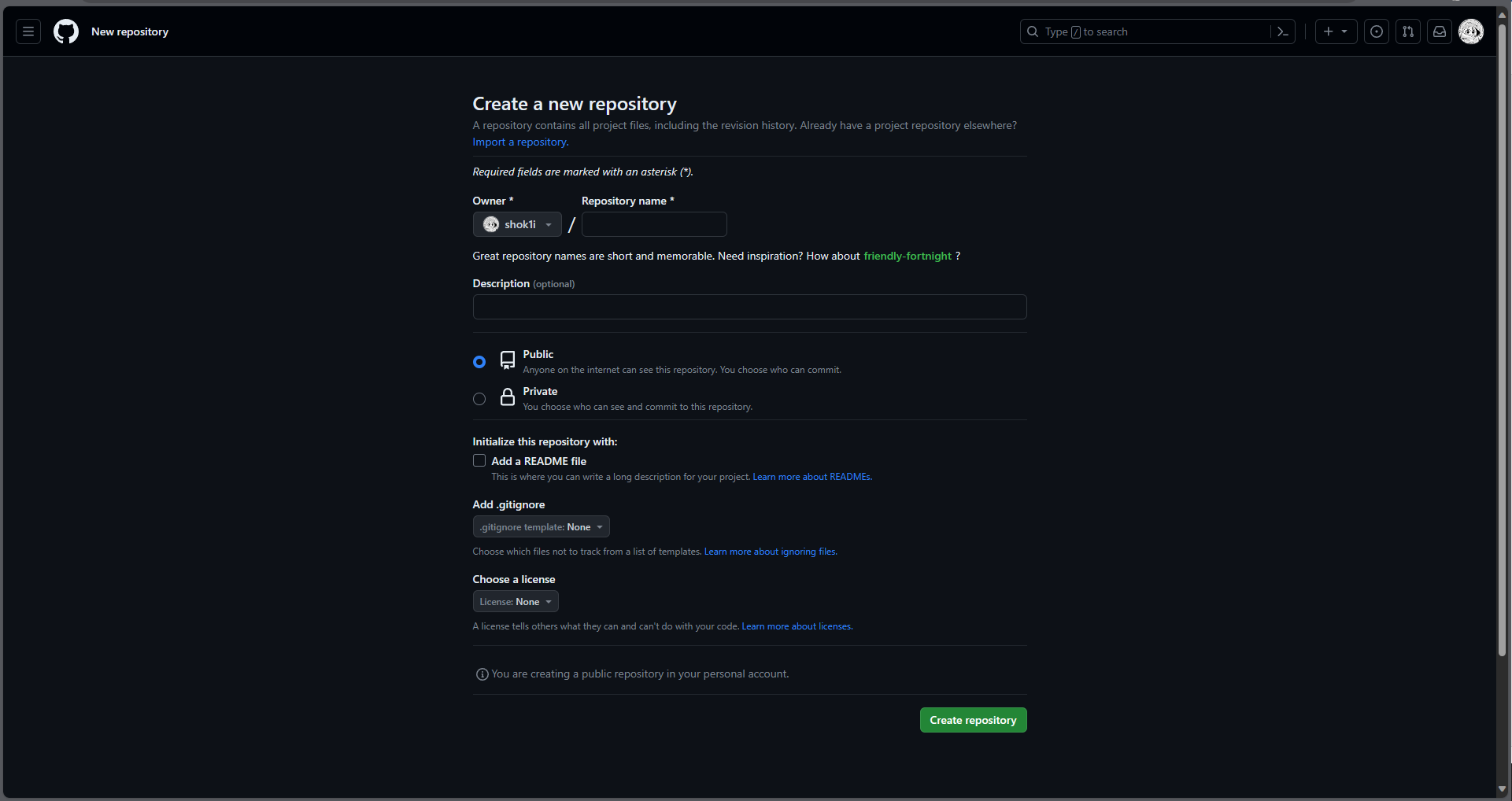


Рисунок 1.5 – Страница настройки нового репозитория

Созданный репозиторий предоставлен на рисунке 1.6



Рисунок 1.6 – Страница с созданным Git репозиторием

*Задание 3:*

Написать программу, в результате которой массив чисел создается с помощью инициализации (как в Си) вводится и считается в цикле сумма элементов целочисленного массива, а также среднее арифметическое его элементов результат выводится на экран. Использовать цикл for.

*Решение:* Для решения задачи был создан метод num\_3, в котором был инициализирован массив array, реализован цикл for, а также две переменные одна для хранения суммы чисел, а другая для их среднеарифметического значения. Так же был реализован вывод полученных результатов в консоль. Демонстрация кода предоставлена на рисунке 1.7.

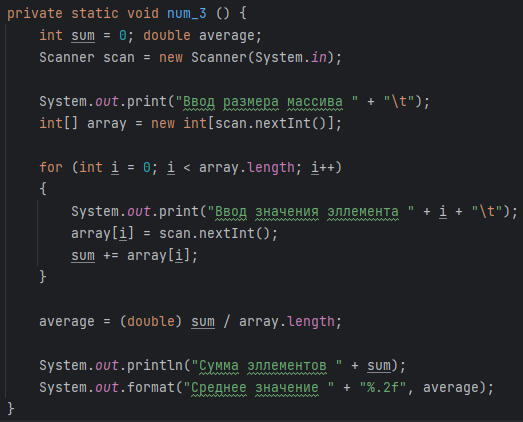


Рисунок 1.7 – Код метода num\_3

Демонстрация работы кода предоставлена на рисунке 1.8.

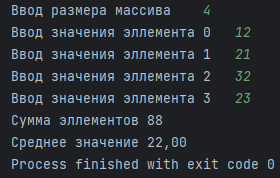


Рисунок 1.8. – Скриншот работы программы

*Задание 4:*

Написать программу, в результате которой массив чисел вводится пользователем с клавиатуры считается сумма элементов целочисленного массива с помощью циклов do while, while, также необходимо найти максимальный и минимальный элемент в массиве, результат выводится на экран.

*Решение:*

Для реализации задачи был создан метод num\_4, в котором был реализована возможность создания массива размерностью, введенной пользователем. С помощью цикла while был реализован ввод каждого элемента в массив пользователем. Затем с помощью цикла do while был создан алгоритм по поиску максимального и минимального значения в массиве. Демонстрация кода предоставлена на рисунке 1.9.

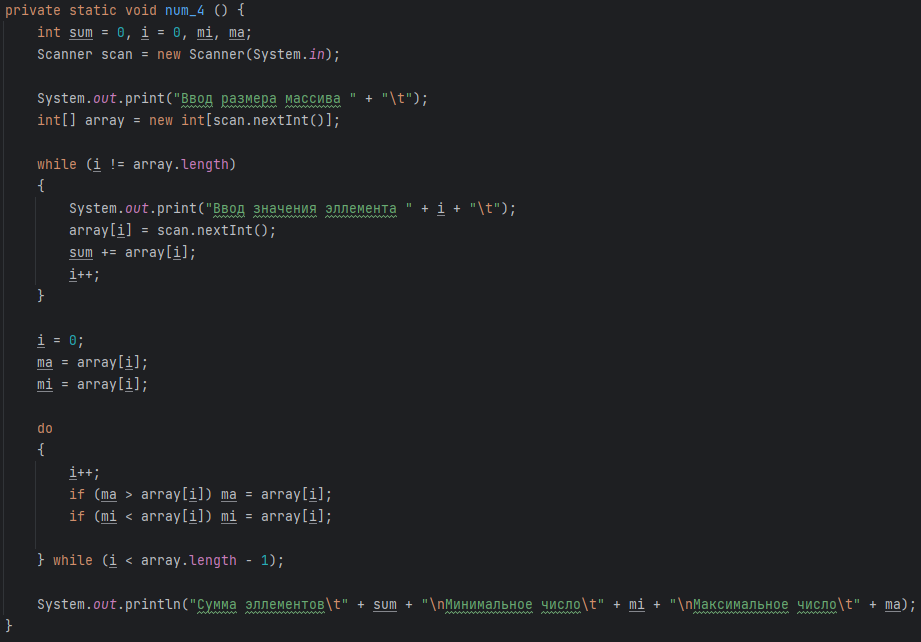


Рисунок 1.9. – Код метода num\_4

Демонстрация работы кода предоставлена на рисунке 1.10.

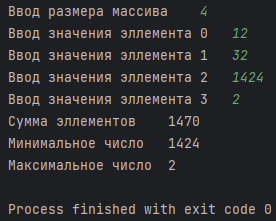


Рисунок 1.10 – Скриншот работы программы

*Задание 5:*

Написать программу, в результате которой выводятся на экран аргументы командной строки в цикле for.

*Решение:*

Для реализации задачи потребуется зайти в раздел (рисунок 1.11) “Current file” и выбрать “Edit Configuration”

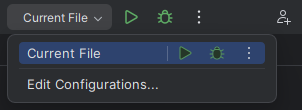


Рисунок 1.11 – раздел “Current file”

В открывшимся окне нажимаем на “Add new…” и выбираем “Application” (рисунок 1.12)

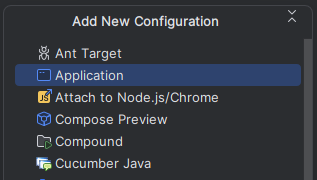


Рисунок 1.12 – Скриншот окна с разделами

В открывшимся окне (рисунок 1.13) вписываем в поле “Program arguments” строку, которая будет передано в параметр args метода main.

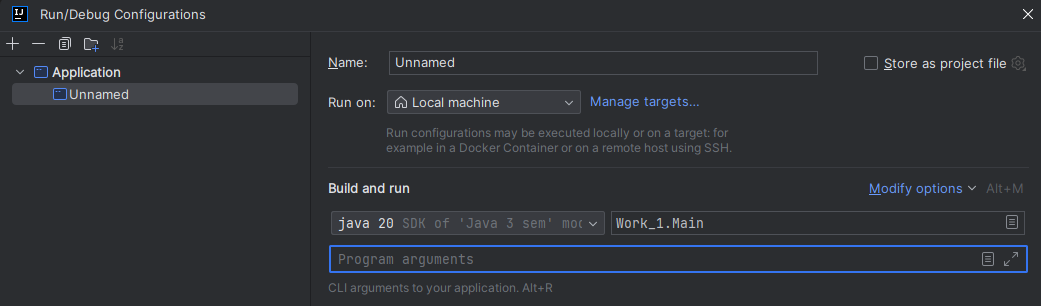


Рисунок 1.13 – Скриншот окна, в котором можно настроить вводимые параметры при запуске программы

Я введу строку “MIREA JAVA LESSONS TASK 5”. Для вывода параметра atgs был реализован метод num\_5, который в качестве входного параметра требует массив строк atgs. Демонстрация кода предоставлена на рисунке 1.14.



Рисунок 1.14 – Код метода num\_5

Демонстрация работы кода предоставлена на рисунке 1.15

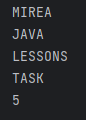


Рисунок 1.15 – Скриншот работы программы

*Задание 6:*

Написать программу, в результате работы которой выводятся на экран первые 10 чисел гармонического ряда (форматировать вывод).

*Решение:*

Для реализации задачи был создан метод num\_6, который выводит первые 10 элементов гармонического ряда используя форматирование. Демонстрация кода предоставлена на рисунке 1.16.

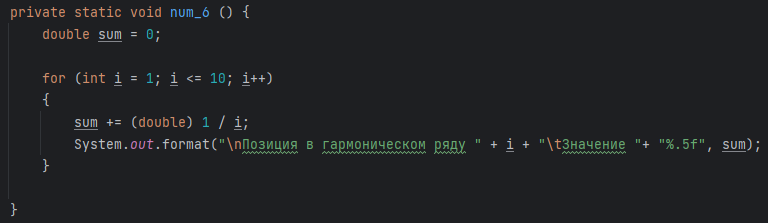


Рисунок 1.16. – Код метода num\_6

Демонстрация работы кода предоставлена на рисунке 1.17.

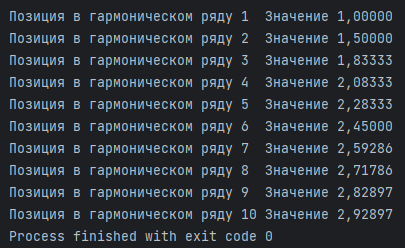


Рисунок 1.17 – Скриншот работы программы

*Задание 7:*

Написать программу, которая с помощью метода класса, вычисляет факториал числа (использовать управляющую конструкцию цикла), проверить работу метода.

*Решение:*

Для реализации задачи был создан метод num\_7, который для подсчета факториала использует рекурсивный метод factorial с целочисленным входным параметром. Демонстрация кода предоставлена на рисунке 1.17.

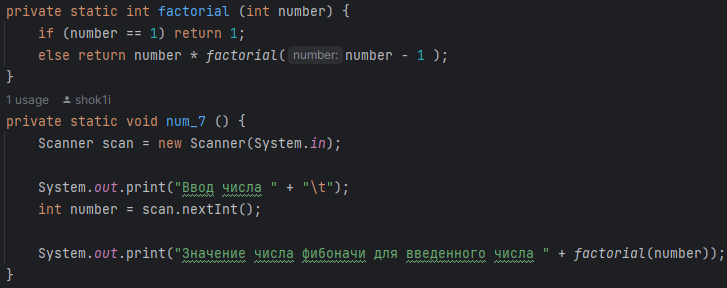


Рисунок 1.17. – Код метода num\_7

Демонстрация работы кода предоставлена на рисунке 1.18.

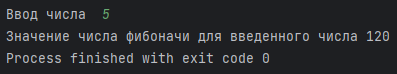


Рисунок 1.18 – Скриншот работы программы

*Задание 8:*

Результаты выполнения практической работы залить через IDE в свой репозитарий и продемонстрировать преподавателю.

*Решение:*

Для реализации задачи я воспользовался программой “Fork” для загрузки файлов в созданный мною репозиторий на GitHub. Результат загрузки файлов представлен на рисунке 1.19.



Рисунок 1.19 – Демонстрация загруженных фалов в Git репозиторий

**1.3 Вывод по работе**

В ходе выполнения данной практической работы были изучены базовый синтаксис и конструкции в Java получены навыки работы с GitHub, Git репозиториями и средой разработки IntelliJ IDEA.

# 2 Практическая работа №2

**2.1 Цель работы**

Изучить основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы. А также изучить и понять, как работают интерфейсы и наследование в языке программирования Java.

**2.2 Теоретическое введение**

Для начала разберем, что такое модификаторы доступа в Java. В Java существуют следующие модификаторы доступа:

1. private: данные класса доступны только внутри класса;
2. protected: данные класса доступны внутри пакета и в наследниках;
3. public: данные класса доступны всем.

Одним из стандартных подходов при проектировании классов на языке Джава является управление доступом к атрибутам класса через пару методов get и set. Метод get позволяет получить значение поля, set — установить новое значение. Общий принцип именования этих методов (называемых, также, геттером и сеттером).

# 2 Практическая работа №2

**2.1 Цель работы**

Изучить основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы. А также изучить и понять, как работают интерфейсы и наследование в языке программирования Java.

**2.2 Теоретическое введение**

Для начала разберем, что такое модификаторы доступа в Java. В Java существуют следующие модификаторы доступа:

1. private: данные класса доступны только внутри класса;
2. protected: данные класса доступны внутри пакета и в наследниках;
3. public: данные класса доступны всем.

Одним из стандартных подходов при проектировании классов на языке Джава является управление доступом к атрибутам класса через пару методов get и set. Метод get позволяет получить значение поля, set — установить новое значение. Общий принцип именования этих методов (называемых, также, геттером и сеттером).

**2.3 Выполнение лабораторной работы**

*Задание 1:*

По диаграмме класса UML описывающей сущность Автор. Необходимо написать программу, которая состоит из двух классов Author и TestAuthor. Класс Author должен содержать реализацию методов, представленных на диаграмме класса на рисунке 2.1.

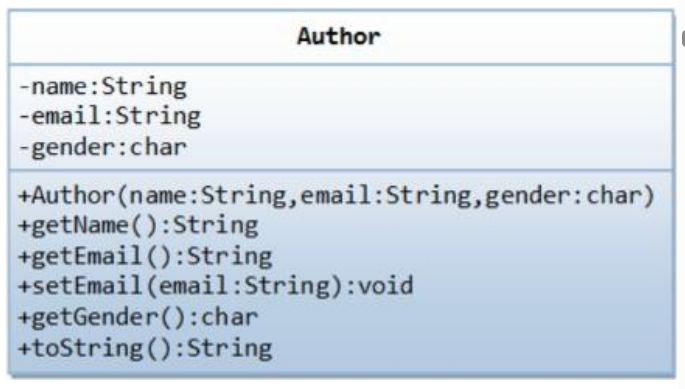


Рисунок 2.1 – Диаграмма класса Author

*Решение:*

Для выполнения данной работы нам необходимо создать два класса Первый класс Author будет содержать реализацию методов необходимых в данной работе, параметризованный конструктор и три поля отвечающих за имя, возраст и пол нашего объекта. И второй класс TestAuthor который будет использоваться для проверки корректности работы методов, написанных в классе Author.

По мимо обычных геттеров и сеттеров нам нужно будет создать метод toString который в качестве возвращаемого значения будет давать нам всю информацию об объекте (имя автора, его пол и почту) в качестве строки.

Демонстрация работы кода предоставлена на рисунке 2.2.

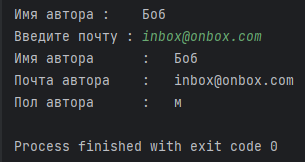


Рисунок 2.2 – Скриншот работы программы

*Задание 2*

По UML диаграмме класса, представленной на рис. 2.3 написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса. Класс Ball должен содержать реализацию методов, представленных на UML. Диаграмма на рисунке описывает сущность Мяч написать программу. Класс Ball моделирует движущийся мяч.

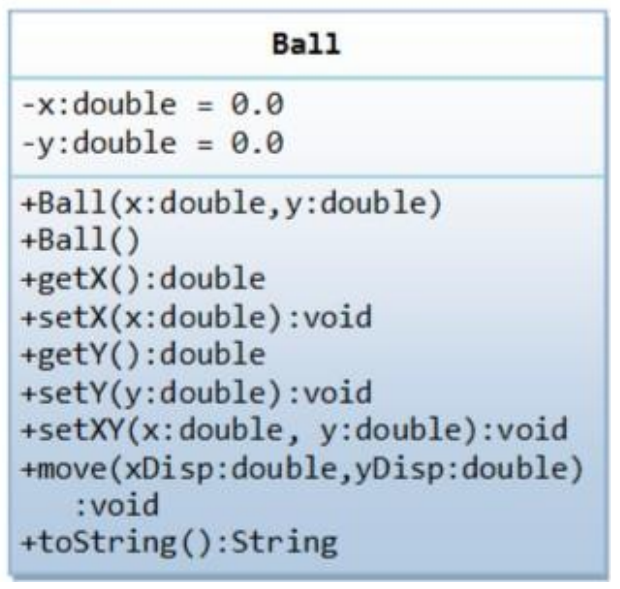


Рисунок 2.3 – Диаграмма класса Ball

*Решение:*

Для выполнения данной работы нам необходимо создать два класса Первый класс Ball будет содержать реализацию методов необходимых в данной работе, параметризованный конструктор и два поля отвечающих за расположение нашего мячика на x/y плоскости. И второй класс BallAuthor который будет использоваться для проверки корректности работы методов, реализованных в классе Ball.

По мимо обычных геттеров и сеттеров нам нужно будет создать метод toString который в качестве возвращаемого значения будет давать нам всю информацию об объекте (расположение мячика на плоскости x/y) в качестве строки. Метод setXY который позволяет задать нашему мячу сразу две координаты в пространстве и метод move, позволяющий передвинуть наш мячик на заданное расстояние.

Демонстрация работы кода предоставлена на рисунке 2.4.

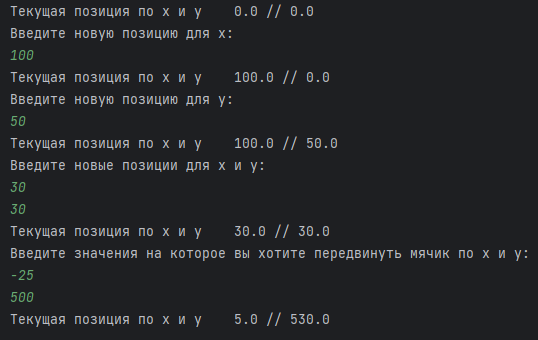


Рисунок 2.4 – Скриншот работы программы

*Задание 3:*

Создать класс точка Point, описывающий точку на плоскости. Создать Circle класс, в котором одно поле представляет точку – центр окружности, и добавить другие свойства, позволяющие задать точку на плоскости. Создать третий класс Tester который использует для хранения объектов массив объектов Circle и второе поле количество элементов в массиве.

*Решение:*

Для решения данного задания нам понадобиться создать три класса, каждый из которых будет иметь свой функционал. Так класс Point будет содержать свойства присущие точке на плоскости x/y, т.е. иметь поля отвечающие за координаты точки и методов позволяющий их задать или изменить. Класс Circle будет иметь в себе поле отвечающие за центр окружности и методы позволяющие изменять координаты этой точки. А также последний класс, класс Tester позволяющий проверить корректность работы программы.

Демонстрация работы кода предоставлена на рисунке 2.5.

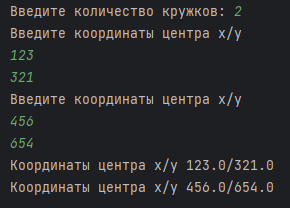


Рисунок 2.5 – Скриншот работы программы

*Задание 4:*

Разработайте класс Shop для, реализуйте методы добавления и удаления компьютеров в магазине, добавьте метод поиска в магазине компьютера, нужного пользователю. Протестируйте работу созданных классов. Данные для заполнения массива компьютеров вводятся с клавиатуры пользователем. Для этого реализуйте интерфейс.

*Решение:*

Для решения данной задачи нам потребуется создать класс Shop который будет содержать контейнер типа ArrayList Pc\_name c данными типа String. Также необходимо будет создать методы, отвечающие за поиск и удаление элемента из контейнера. Также необходимо реализовать интерфейс для последующего ввода с клавиатуры и добавления введённого элемента в контейнер. Также нам необходимо будет создать еще один класс для последующий проверки корректности работы программы Start.

Демонстрация работы кода предоставлена на рисунке 2.6.

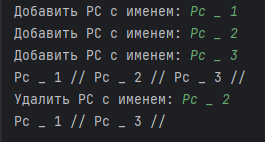


Рисунок 2.6 – Скриншот работы программы

*Задание 5:*

Разработайте и реализуйте класс Dog (Собака), поля класса описывают кличку и возраст собаки. Необходимо выполнить следующие действия: определить конструктор собаки, чтобы принять и инициализировать данные экземпляра., включить стандартные методы (аксессоры) для получения и установки для имени и возраста, включить метод для перевода возраста собаки в “человеческий” возраст (возраст семь раз собаки), включите метод ToString, который возвращает описание экземпляра собаки в виде строки. Создание класса тестера под названием DogKennel, реализует массив собак и основной метод этого класса позволяет добавить в него несколько объектов собаки.

*Решение:*

Для решения данной задачи нам потребуется создать класс Dog который включает в себя поля строкового типа Name и целочисленное Age. Помимо сеттеров и геттеров нам понадобиться реализовать методы для перевода собачьего возраста в человечий то есть умножение возраста собаки на семь и метод ToString который в качестве возвращаемого значения будет давать нам всю информацию об объекте (кличку собаки и ее возраст) в качестве строки. Также класс DogKennel который будет отвечать за проверку корректности отработки программы.

Демонстрация работы кода предоставлена на рисунке 2.7.

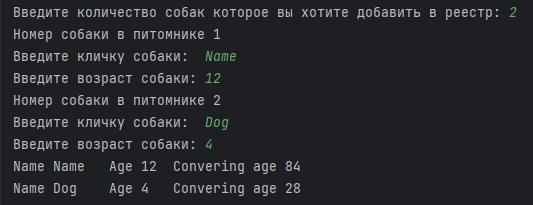


Рисунок 2.7 – Скриншот работы программы

*Задание 6:*

Создать класс, описывающий модель окружности (Circle). В классе должны быть описаны нужные свойства окружности и методы для получения и изменения этих свойств. Добавить методы для расчета площади круга и длины окружности, а также метод позволяющий сравнивать две окружности. При помощи класса CircleTest, содержащего статический метод main(String[] args), протестировать работу класса Circle.

*Решение:*

Для решения данной задачи нам потребуется создать два класса. Класс Circle который имеет поле с типом расширенная плывущая точка. В этом классе помимо сеттера и геттера нужно реализовать методы, которые будут считать длину, площадь и периметр окружность и метод для сравнения двух окружностей. Класс CircleTest который будет отвечать за проверку корректности отработки программы.

Демонстрация работы кода предоставлена на рисунке 2.9.

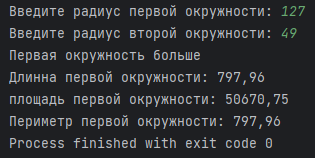


Рисунок 2.9 – Скриншот работы программы

*Задание 7:*

Создать класс, описывающий книгу (Book). В классе должны быть описаны нужные свойства книги (автор, название, год написания и т. д.) и методы для получения, изменения этих свойств. Протестировать работу класса в классе BookTest, содержащим метод статический main(String[] args). Создать класс книжная полка, в котором поля данных класса это массив объектов типа книги (Book, и количество книг на книжной полке. Написать методы класса, которые возвращают книги с самым поздним и самым ранним сроком издания. Написать метод класса, позволяющий расставить книги на книжной полке в порядке возрастания года выпуска. Используйте реализацию отношений композиция классов.

*Решение:*

Для решения потребуется создать три класса:

Book включает в себя поля с информацией о книге (имя автора, название книги, название издателя, и год выпуска). А также помимо сеттеров и геттеров включает в себя метод BookInfo который в качестве возвращаемого значения будет давать нам всю информацию об объекте (имя автора, название книги, название издателя, и год выпуска) в качестве строки.

BookShelf включает в себя поле с массивом типа Book и целочисленное значение отвечающие за количество книг на полке. А также помимо обычных геттеров и сеттеров включает в себя методы для получения самой новой и старой книги, и для сортировки книг по дате выпуска на полке.

BookTest который будет отвечать за проверку корректности отработки программы.

Демонстрация работы кода предоставлена на рисунке 2.8.

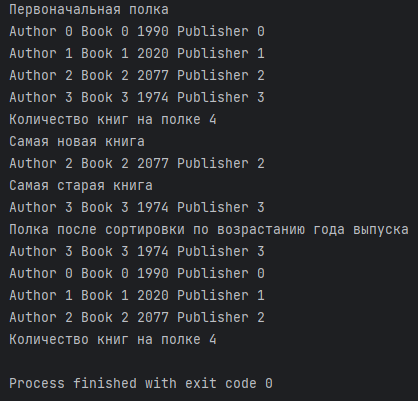


Рисунок 2.8 – Скриншот работы программы

*Задание 8:*

Напишите программу, которая меняет местами элементы

одномерного массива из String в обратном порядке. Не используйте

дополнительный массив для хранения результатов.

*Решение:*

Для решения данной задачи нам необходимо создать массив строк и через цикл for поменять значения в строке. Демонстрация кода предоставлена на рисунке 2.9.

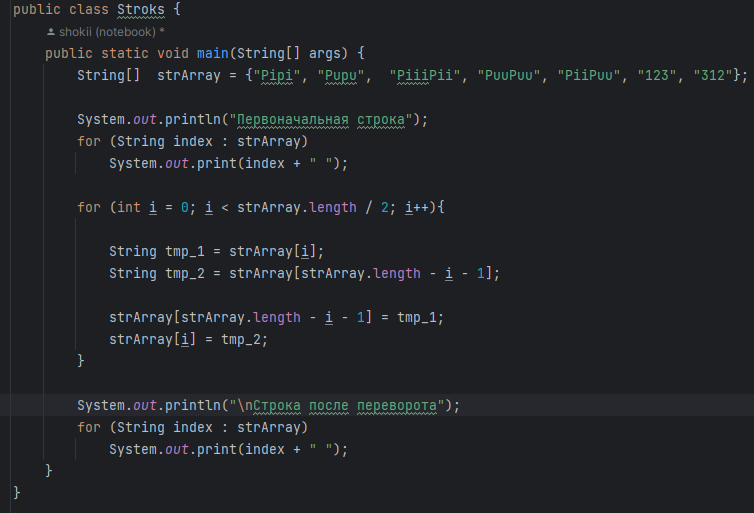


Рисунок 2.9. – Код реализации программы для задания 8

Демонстрация работы кода предоставлена на рисунке 2.10.

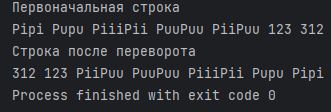


Рисунок 2.10 – Скриншот работы программы

*Задание 9:*

Напишите программу Poker.java, которая должна имитировать раздачу карт для игры в покер. Программа получает число n, задаваемое с консоли пользователем, и раздает карты на n игроков (по 5 карт каждому) из перетасованной колоды. Разделяйте пять карт, выданных каждому игроку, пустой строкой

*Решение:*

Для решения данной задачи нам потребуется создать 5 классов:

Cards включает в себя поля для масти и значения карты и методы для их получения.

CardHolder включает в себя контейнер типа ArrayList c типом данных Cards и методы для добавления и удаления определенной карты из колоды.

Player включает в себя массив типа Cards и методы для взятия двух карт в руки и метод для получения информации о картах находящихся у игрока в руках.

Croupier включает в себя массив типа Cards и методы для взятия пяти карт в руки и метод для получения информации о картах находящихся у крупье в руках.

Poker который будет отвечать за проверку корректности отработки программы.

Демонстрация работы кода предоставлена на рисунке 2.11.

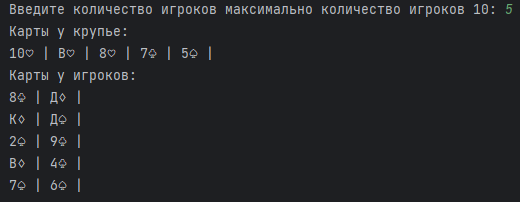


Рисунок 2.11 – Скриншот работы программы

*Задание 10:*

Напишите программу HowMany.java, которая определит, сколько слов Вы ввели с консоли

*Решение:*

Для решения нужно создать класс HowMany которая будет считать количество слов, введённое в консоль.

Демонстрация работы кода предоставлена на рисунке 2.12.

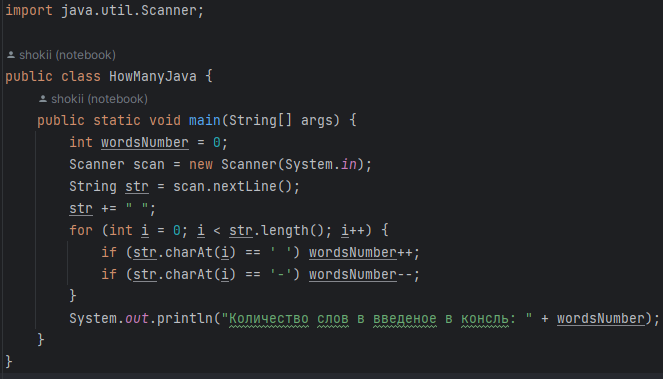


Рисунок 2.12 - Код реализации программы для задания 10

Демонстрация кода предоставлена на рисунке 2.13.

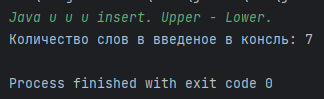


Рисунок 2.13 – Скриншот работы программы

**2.4 Вывод по работе**

Были изучены основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы. А также было изучено, как работают интерфейсы и наследование в языке программирования Java.

# 3 Практическая работа №3

**2.1 Цель работы**

Освоить на практике работу с абстрактными классами и наследованием на Java.

**2.2 Теоретическое введение**

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы при определении помечаются ключевым словом abstract.

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип. Он состоит только из объявления и не имеет тела:

abstract void yourMethod();

По сути, мы создаём шаблон метода. Например, можно создать абстрактный метод для вычисления площади фигуры в абстрактном классе Фигура. А все другие производные классы от главного класса могут уже реализовать свой код для готового метода. Ведь площадь у прямоугольника и треугольника вычисляется по разным алгоритмам и универсального метода не существует.

Если вы объявляете класс, производный от абстрактного класса, но хотите иметь возможность создания объектов нового типа, вам придётся предоставить определения для всех абстрактных методов базового класса. Если этого не сделать, производный класс тоже останется абстрактным, и компилятор заставит пометить новый класс ключевым словом abstract.

Абстрактный класс не может содержать какие-либо объекты, а также абстрактные конструкторы и абстрактные статические методы. Любой подкласс абстрактного класса должен либо реализовать все абстрактные методы суперкласса, либо сам быть объявлен абстрактным.

**2.3 Выполнение лабораторной работы**

*Задание 1:*

Создайте абстрактный родительский суперкласс Shape и его дочерние классы (подклассы).

*Решение:*

Были созданы абстрактный класс Shape и его дочерние публичные классы Rectangle, Circle, Square (рисунок 3.1).

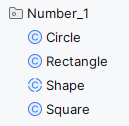
**

Рисунок 3.1 – Созданные нами классы в проекте

*Задание 2:*

Перепишите суперкласс Shape и его подклассы, так как это представлено на диаграмме Circle, Rectangle and Square рисунка 3.2.

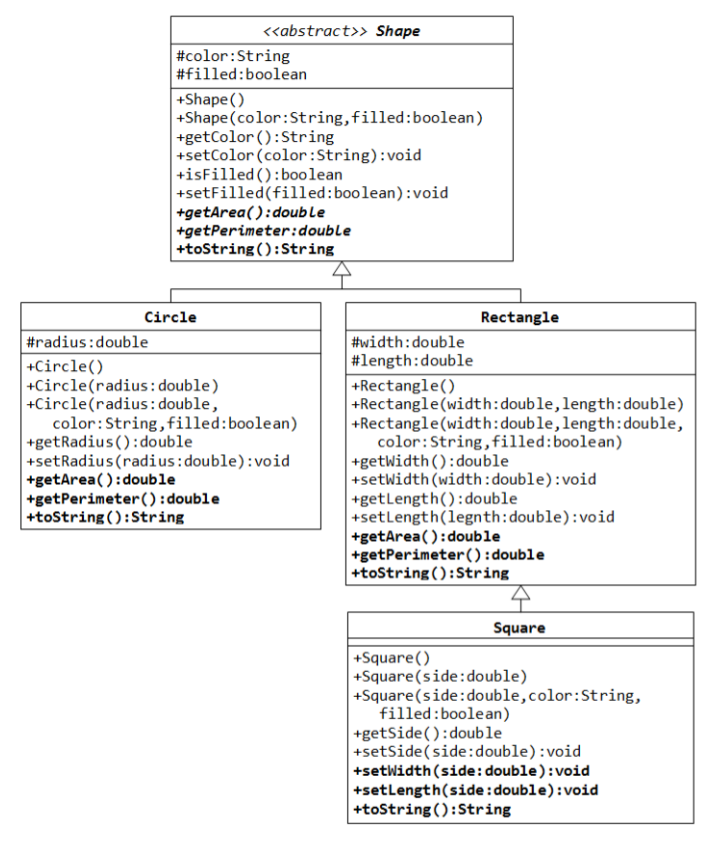


Рисунок 3.2 - Диаграмма суперкласса Shape

*Решение:*

Для выполнения данной задачи нам потребуется переписать классы из предыдущей задачи и реализовать в каждом из них поля и методы, описанные в диаграмме, показанной на рисунке 3.2.

Методы getArea и getPerimetr класа Shape мы должны сделать абстрактными.

Поля у каждого класса мы должны сделать типом protected.

*Задание 3:*

Вам нужно написать тестовый класс, чтобы самостоятельно это проверить, необходимо объяснить полученные результаты и связать их с понятием ООП - полиморфизм. Некоторые объявления могут вызвать ошибки компиляции. Объясните полученные ошибки, если таковые имеются.

*Решение:*

Для выполнения данной задачи мы должны создать новый класс – класс Tester. Благодаря которому мы сможем убедиться в работоспособности программы.

Благодаря проведенным тестом мы смогли выявить пять ошибок.

System.out.println(s1.getRadius()); - Родительский абстрактный класс Shape не содержит метод getRadius() класса Circle

Shape s2 = new Shape(); - Shape это абстрактный класс именно поэтому мы и не можем его инициализировать.

System.out.println(s3.getLength()); - Родительский абстрактный класс Shape не содержит метод getLength () класса Square

System.out.println(s4.getSide()); - Абстрактный класс Shape не содержит метод getSide () класса Circle

System.out.println(r2.getSide()); - Родительский класс Rectangle не содержит метод getSide () наследника

Результат отработки программы не включая методов, которые вызывают ошибки предоставлены на рисунке 3.3

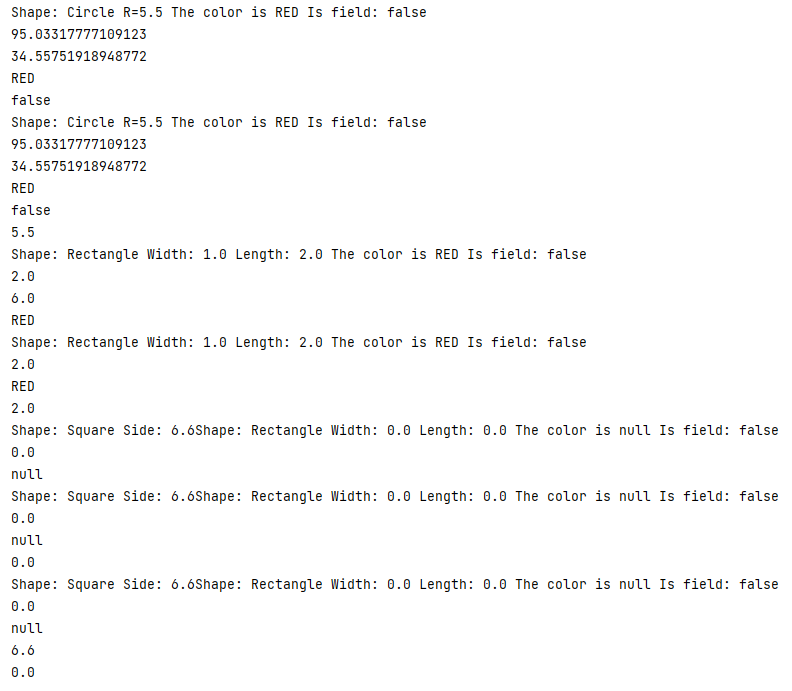


Рисунок 3.3 – Скриншот отработки программы

Задание 4:

Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable (рисунок 3.4).

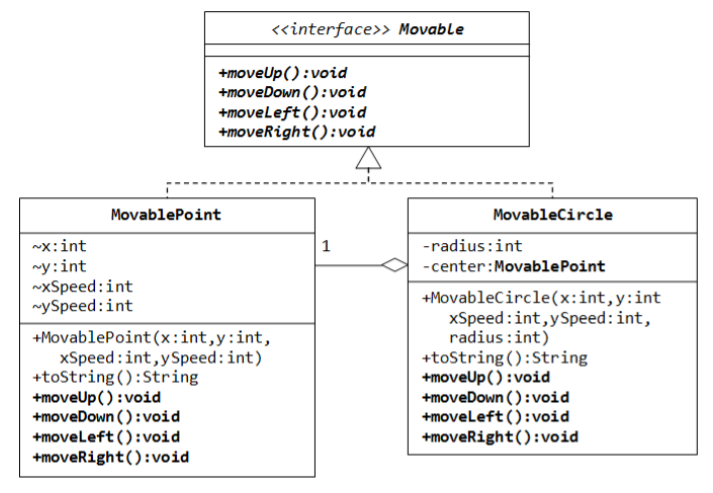


Рисунок 3.4 – Диаграмма реализации интерфейса Movable

*Решение:*

Для реализации данной задачи были написаны один интерфейс и два класса. В классах были реализованы поля типа private и так же методы, находящиеся в диаграмме. В интерфейсе были написаны четыре метода moveUP, moveDOWN, moveLEFT и moveRIGHT. Также был реализован класс Tester, который был использован для проверки корректности работы написанных методов.

Результат отработки программы показан на рисунке 3.5.

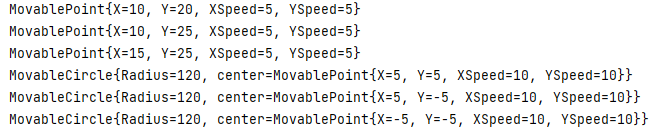


Рисунок 3.5 – Скриншот отработки программы

*Задание 5:*

Напишите новый класс MovableRectangle (рисунок 3.6) (движущийся прямоугольник). Его можно представить как две движущиеся точки MovablePoints (представляющих верхняя левая и нижняя правая точки) иреализующие интерфейс Movable. Убедитесь, что две точки имеет одну и ту же скорость (нужен метод это проверяющий).

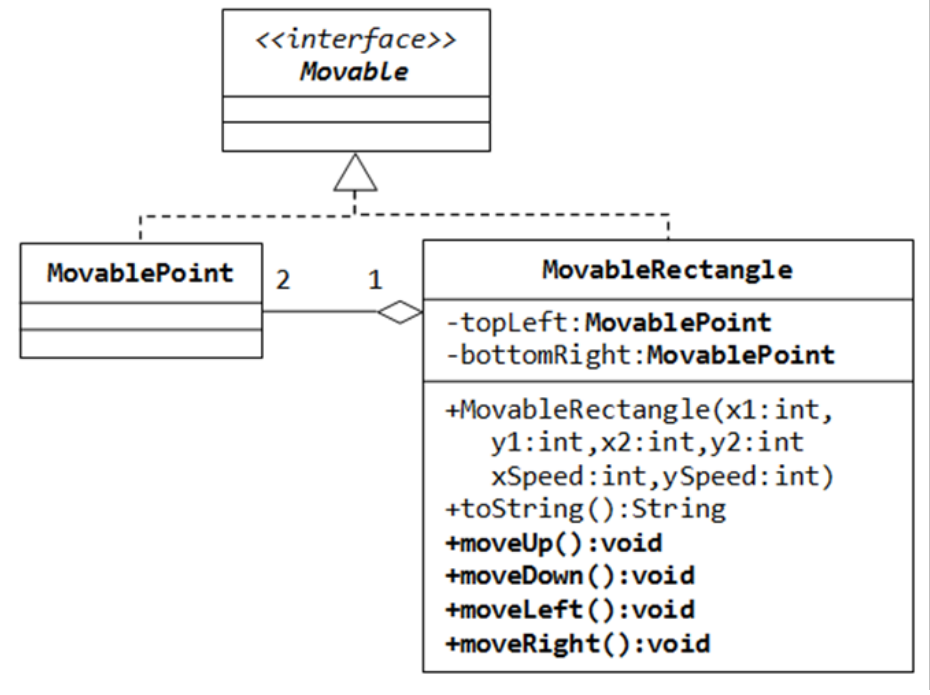
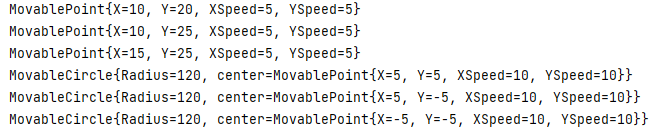


Рисунок 3.6 – Диаграмма класса MovableRectangle

*Решение:*

Для кода, реализованного в предыдущем задании нам, нужно добавить новый класс MovableRectangle и переработать класс Tester добавив код для проверки методов написанных в созадачном в классе.

Результат отработки программы показан на рисунке 3.7.

Рисунок 3.7 – Скриншот отработки программы

**3.3 Вывод по работе**

В ходе выполнения данной практической работы были изучены и применены правила, по которым работают интерфейсы и наследование в языке программирования Java.

# 4 Практическая работа №4

**4.1 Цель работы**

Введение в событийное программирование на языке Java.

**4.2 Теоретическое введение**

Данная практическая работа посвящена закреплению практических навыков по созданию приложений на Java c использованием следующих элементов GUI:

1. Текстовые поля и области ввода текста;
2. Менеджеры компоновки компонентов;
3. Слушатель мыши;
4. Создание меню.

Text Fields - текстовое поле или поля для ввода текста (можно ввести только одну строку). Примерами текстовых полей являются поля для ввода логина и пароля, например, используемые, при входе в электронную почту.

Пример создания объекта класса JTextField:

JTextField jta = new JTextField (10);

В параметрах конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст веденный в поле JText может быть возвращен с помощью метода getText(). Также в поле можно записать новое значение с помощью метода setText(String s). Как и у других компонентов, мы можем изменять цвет и шрифт текста в текстовом поле.

**4.3 Выполнение лабораторной работы**

*Задание 1:*

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид. Создайте JFrame приложение, у которого есть следующие компоненты GUI:

1. одна кнопка JButton labeled “AC Milan”
2. другая JButton подписана “Real Madrid”
3. надпись JLabel содержит текст “Result: 0 X 0”
4. надпись JLabel содержит текст “Last Scorer: N/A”
5. надпись Label содержит текст “Winner: DRAW”;

Всякий раз, когда пользователь нажимает на кнопку AC Milan, результат будет увеличиваться для Милана, сначала 1 X 0, затем 2 X 0 и так далее. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. Если пользователь нажимает кнопку для команды Мадрид, то счет приписывается ей. Победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.

*Решение:*

Для работы я реализовал 3 класса:

1. Класс MainFrame – в котором реализуется основной код программы (создание объектов размещение GUI-элементов);
2. Класс AL\_MILAN – в котором мы реализовали конструктор и метод для обработки нажатий мышкой на кнопку;
3. Класс REAL\_MADRID – в котором мы реализовали конструктор и метод для обработки нажатий мышкой на кнопку.

Реализация одного из классов предоставлена на рисунке 4.1.

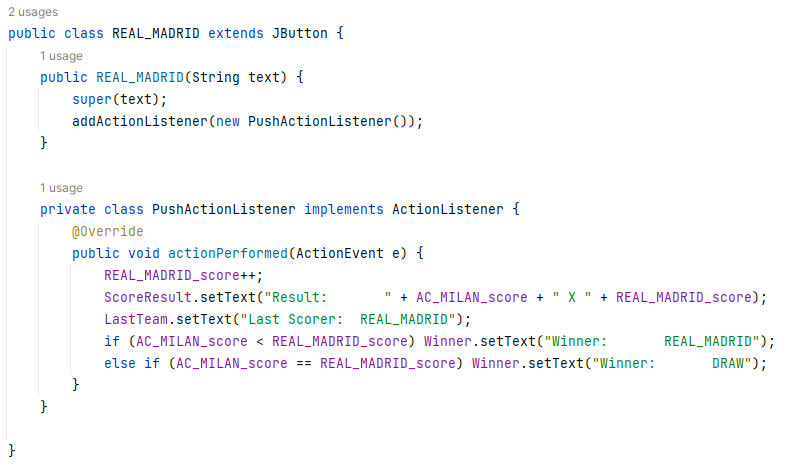


Рисунок 4.1 – Скриншот реализации класса REAL\_MADRID

Результат отработки программы вывел данное окно предоставлена на рисунке 4.2.

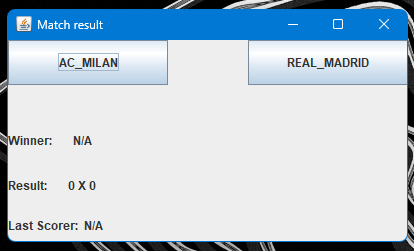


Рисунок 4.2 – Скриншот появившегося окна после запуска программы

**4.3 Вывод по работе**

В ходе данной практической работы были получены навыки работы с событийном программированием в Java также было подмечено что, оно позволяет создавать интерактивные и отзывчивые приложения.

Благодаря событийному программированию разработчики могут создавать компоненты графического интерфейса, регистрировать различные события и определять конкретные действия, которые должны выполняться при наступлении этих событий. Это позволяет разработчикам создавать удобные приложения, реагирующие на действия, выполняемые пользователя в реальном времени.

# 5 Практическая работа №5

**5.1 Цель работы**

Разработка и программирование рекурсивных алгоритмов на языке Java.

**5.2 Теоретическое введение**

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Так же, как и у перебора (цикла) у рекурсии должно быть условие остановки — базовый случай (иначе также, как и цикл, рекурсия будет работать вечно). Это условие и является тем случаем, к которому рекурсия идет (шаг рекурсии). При каждом шаге вызывается рекурсивная функция до тех пор, пока при следующем вызове не сработает базовое условие и не произойдет остановка рекурсии (а точнее возврат к последнему вызову функции). Всё решение сводится к поиску решения для базового случая. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

Итак, рекурсивная функция состоит из:

1. условие остановки или же базового случая или условия;
2. условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам.

**5.3 Выполнение лабораторной работы**

*Задание 12:*

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите все нечетные числа из этой последовательности, сохраняя их порядок. В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 5.1.

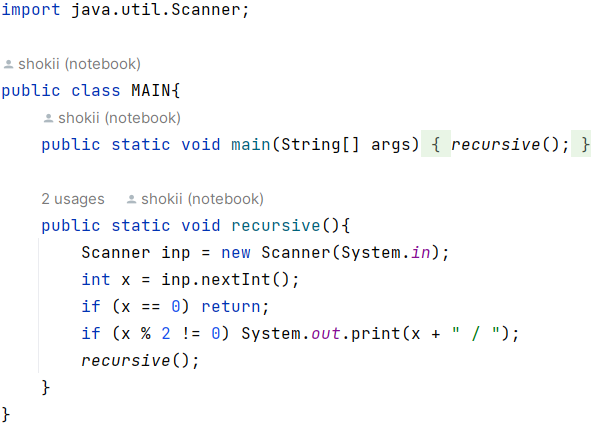


Рисунок 5.1 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 5.2.

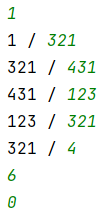


Рисунок 5.2 – Скриншот результата отработки программы

*Задание 13:*

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо. 64 В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 5.3.

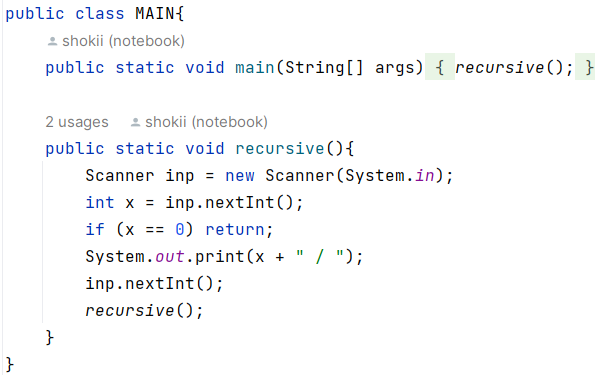


Рисунок 5.3 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 5.4.

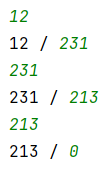


Рисунок 5.4 – Скриншот результата отработки программы

*Задание 14:*

Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 5.5.

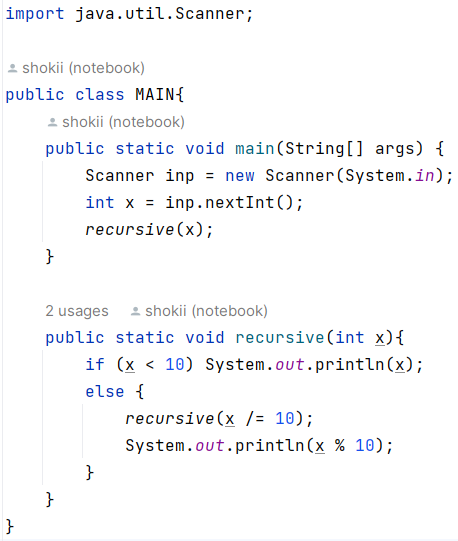


Рисунок 5.5 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 5.6.

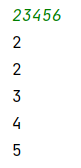


Рисунок 5.6 – Скриншот результата отработки программы

*Задание 15:*

Дано натуральное число N. Выведите все его цифры по одной, в обратном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 5.7.

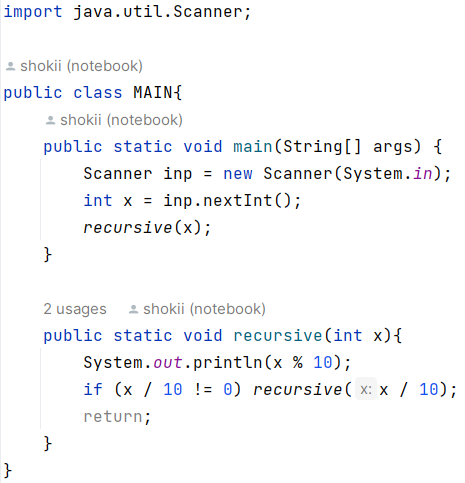


Рисунок 5.7 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 5.8.

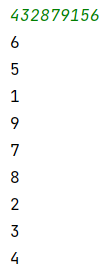


Рисунок 5.8 – Скриншот результата отработки программы

**5.3 Вывод по работе**

В ходе данной практической работы были получены навыки работы с рекурсивными алгоритмами на языке Java.

Рекурсивное программирование помогает оптимизировать программу создать простые решения для трудных задач, используя присущую методу особенность, а именно вызов самого себя. Однако очень важно обеспечить правильное условие для завершения рекурсивного вызов, чтобы избежать бесконечной рекурсии.

# 6 Практическая работа №6

**6.1 Цель работы**

Цель данной работы - изучить понятие наследования, и научиться реализовывать наследование в Java.

**6.2 Теоретическое введение**

Одним из ключевых аспектов объектно-ориентированного программирования является наследование. С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого.

*Абстрактные классы*

Кроме обычных классов в Java есть абстрактные классы. Абстрактный класс похож на обычный класс. В абстрактном классе также можно определить поля и методы, в то же время нельзя создать объект или экземпляр абстрактного класса. Абстрактные классы призваны предоставлять базовый функционал для классов- наследников. А производные классы уже реализуют этот функционал.

**6.3 Выполнение лабораторной работы**

*Задание 1:*

Создать абстрактный класс, описывающий посуду (Dish). С помощью наследования реализовать различные виды посуды, имеющие свои свойства и методы. Протестировать работу классов.

*Решение:*

Для реализации данной задачи нам потребуется реализовать абстрактный класс Dish с двумя полями строкового типа. Одно поле будет отвечать за тип посуды, а второе за ее цвет. Реализация данного класса предоставлена на рисунке 6.1.

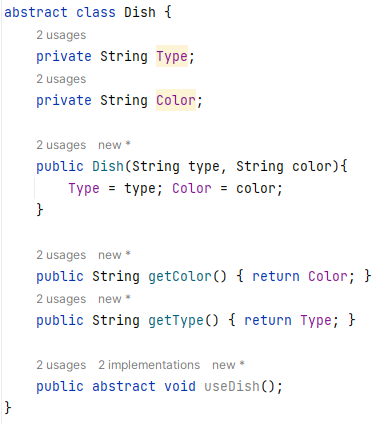


Рисунок 6.1 – Реализация абстрактного класса Dish

Также были реализованы два класса, наследуемые от Dish – это класс Cup и класс Plate. Реализация которых показана на рисунке 6.2.

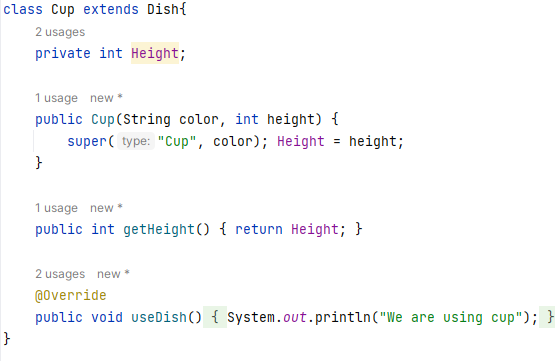
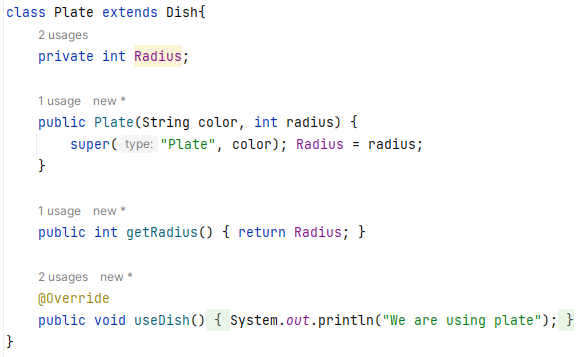


Рисунок 6.2 – Реализация классов Cup и Plate

Результат отработки написанной пррограммы предоставлен на скриншоте 6.3.

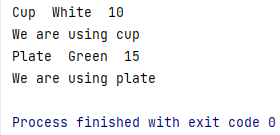


Рисунок 6.3 – Скриншот результата отработки программы

*Задание 2:*

Создать абстрактный класс, описывающий собак (Dog). С помощью наследования реализовать различные породы собак. Протестировать работу классов.

*Решение:*

Для реализации данной задачи нам потребуется реализовать абстрактный класс Dog с двумя полями строкового типа. Одно поле будет отвечать за породу собаки, а второе за ее возраст. Реализация данного класса предоставлена на рисунке 6.4.

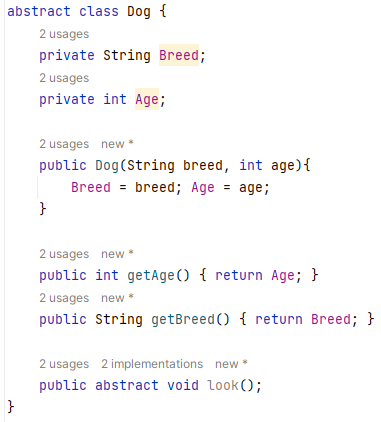


Рисунок 6.4 – Реализация абстрактного класса Dog

Также были реализованы два класса, наследуемые от Dog – это класс Bulldog и класс Terrier. Реализация которых показана на рисунке 6.5.

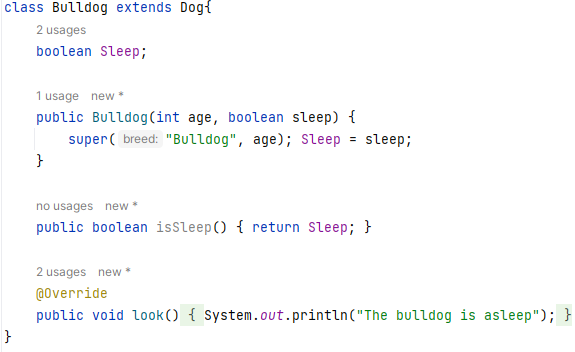
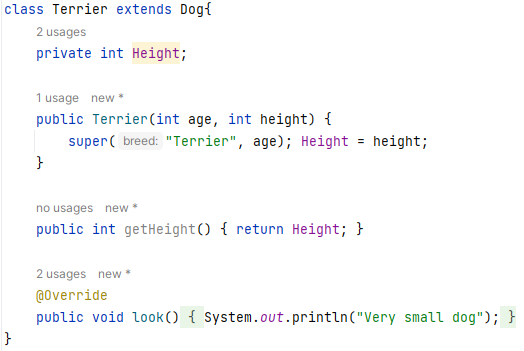
 

Рисунок 6.5 – Реализация классов Cup и Plate

Результат отработки написанной пррограммы предоставлен на скриншоте 6.6.

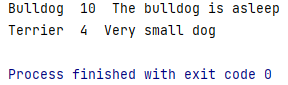


Рисунок 6.6 – Скриншот результата отработки программы

*Задание 3:*

Создать абстрактный класс, описывающий мебель. С помощью наследования реализовать различные виды мебели. Также создать класс FurnitureShop, моделирующий магазин мебели. Протестировать работу классов.

*Решение:*

Для реализации данной задачи нам потребуется реализовать абстрактный класс Furniture с тремя полями. Одно поле строкового типа будет отвечать за тип мебели, а два поля целочисленного значения будут отвечать за вес и цену. Реализация данного класса предоставлена на рисунке 6.7.

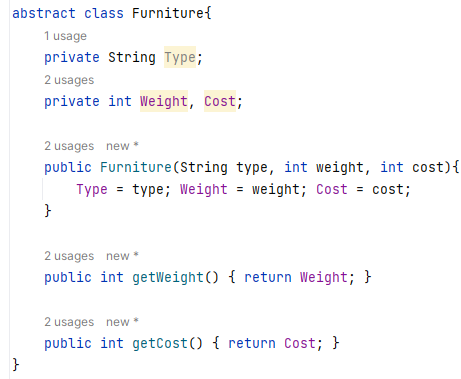


Рисунок 6.7 – Реализация абстрактного класса Furniture

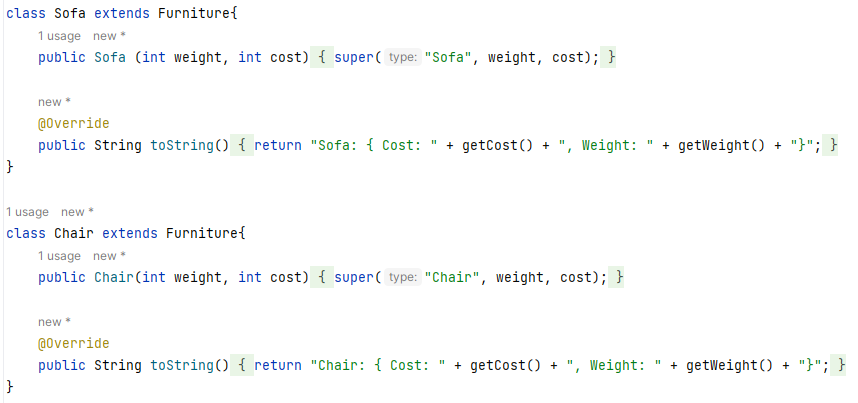
Также были реализованы два класса, наследуемые от Furniture – это класс Sofa и класс Chair. Реализация которых показана на рисунке 6.8. 

Рисунок 6.8 – Реализация классов Sofa и Chair

Результат отработки написанной программы предоставлен на скриншоте 6.6.

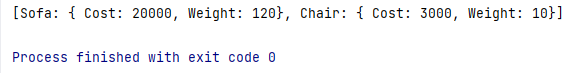


Рисунок 6.9 – Скриншот результата отработки программы

**6.3 Вывод по работе**

# 7 Практическая работа №7

**7.1 Цель работы**

Введение в событийное программирование на языке Java.

**7.2 Теоретическое введение**

Данная практическая работа посвящена закреплению практических навыков по созданию приложений на Java c использованием следующих элементов GUI:

1. Текстовые поля и области ввода текста;

2) Менеджеры компоновки компонентов;

1. Слушатель мыши;
2. Создание меню.

Text Fields - текстовое поле или поля для ввода текста (можно ввести только одну строку). Примерами текстовых полей являются поля для ввода логина и пароля, например, используемые, при входе в электронную почту.

Пример создания объекта класса JTextField:

JTextField jta = new JTextField (10);

В параметрах конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст веденный в поле JText может быть возвращен с помощью метода getText(). Также в поле можно записать новое значение с помощью метода setText(String s). Как и у других компонентов, мы можем изменять цвет и шрифт текста в текстовом поле.

**7.3 Выполнение лабораторной работы**

*Задание 1:*

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид. Создайте JFrame приложение, у которого есть следующие компоненты GUI:

1. одна кнопка JButton labeled “AC Milan”
2. другая JButton подписана “Real Madrid”
3. надпись JLabel содержит текст “Result: 0 X 0”
4. надпись JLabel содержит текст “Last Scorer: N/A”
5. надпись Label содержит текст “Winner: DRAW”;

Всякий раз, когда пользователь нажимает на кнопку AC Milan, результат будет увеличиваться для Милана, сначала 1 X 0, затем 2 X 0 и так далее. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. Если пользователь нажимает кнопку для команды Мадрид, то счет приписывается ей. Победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.

*Решение:*

Для работы я реализовал 3 класса:

1. Класс MainFrame – в котором реализуется основной код программы (создание объектов размещение GUI-элементов);
2. Класс AL\_MILAN – в котором мы реализовали конструктор и метод для обработки нажатий мышкой на кнопку;
3. Класс REAL\_MADRID – в котором мы реализовали конструктор и метод для обработки нажатий мышкой на кнопку.

Реализация одного из классов предоставлена на рисунке 7.1.

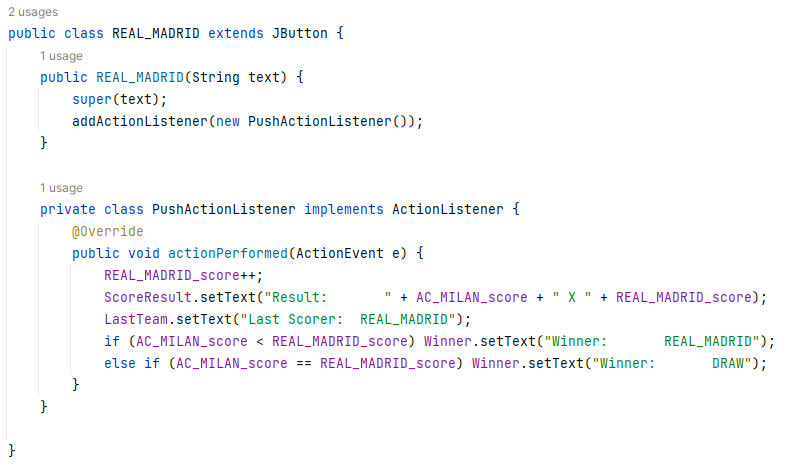


Рисунок 7.1 – Скриншот реализации класса REAL\_MADRID

Результат отработки программы вывел данное окно предоставлена на рисунке 7.2.

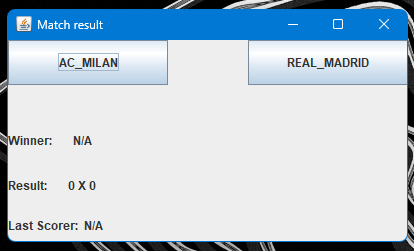


Рисунок 7.2 – Скриншот появившегося окна после запуска программы

**7.3 Вывод по работе**

В ходе данной практической работы были получены навыки работы с событийном программированием в Java также было подмечено что, оно позволяет создавать интерактивные и отзывчивые приложения.

Благодаря событийному программированию разработчики могут создавать компоненты графического интерфейса, регистрировать различные события и определять конкретные действия, которые должны выполняться при наступлении этих событий. Это позволяет разработчикам создавать удобные приложения, реагирующие на действия, выполняемые пользователя в реальном времени.

# 8 Практическая работа №8

**8.1 Цель работы**

Научиться создавать графический интерфейс пользователя, освоить на практике работу с различными объектами для создания ГИП, менеджерами размещения компонентов.

**8.2 Теоретическое введение**

Теоретические сведения указаны в практической работе №7.

**8.3 Выполнение лабораторной работы**

*Задание 1:*

Создать окно, нарисовать в нем 20 случайных фигур, случайного цвета. Классы фигур должны наследоваться от абстрактного класса Shape, в котором описаны свойства фигуры: цвет, позиция.

*Решение:*

Для реализации поставленной задачи нам необходимо создать абстрактный класс Shape. Его реализация предоставлена на рисунке 8.1.



Рисунок 8.1 – Реализация абстрактного класса Shape

Также были реализованы два класса, которые являются наследниками класса Shape. Класс Rectangle отвечает за создание прямоугольников, а класс Circle отвечает за создание окружностей.

Результат отработки программы предоставлен на рисунке 8.2.

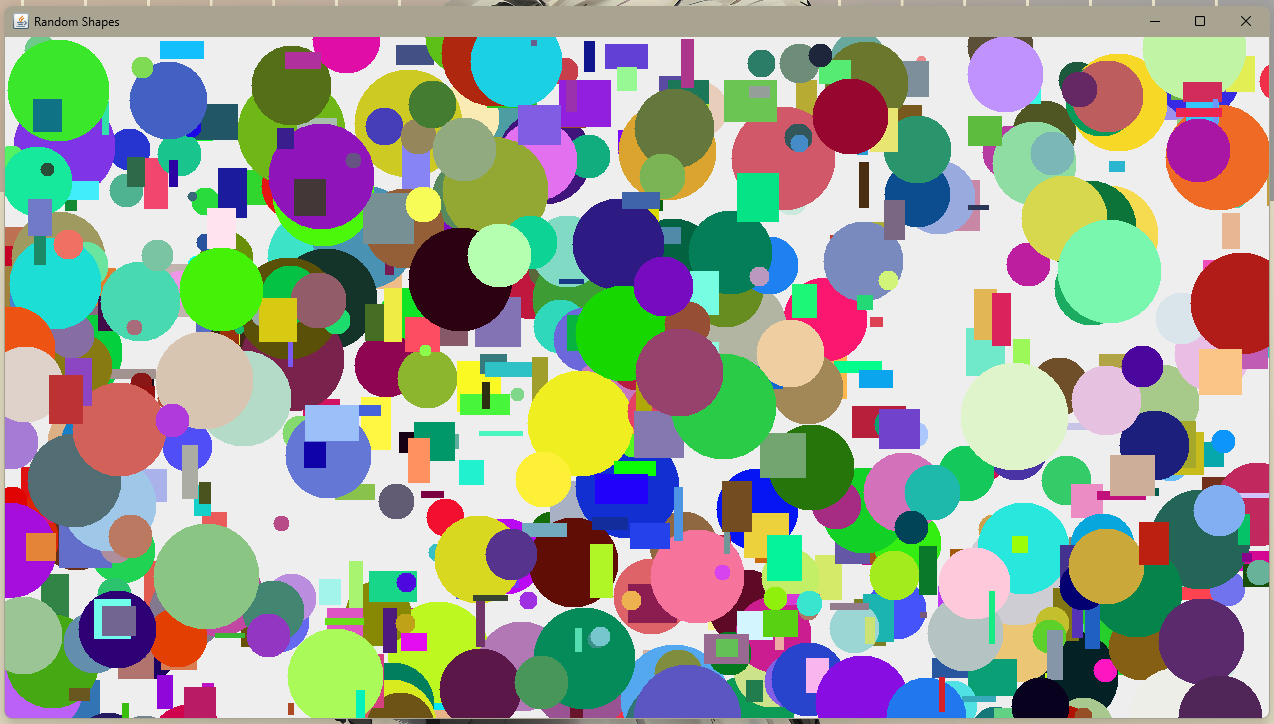


Рисунок 8.2 – Скриншот появившегося окна после запуска программы

*Задание 2:*

Создать окно, отобразить в нем картинку, путь к которой указан в аргументах командной строки.

*Решение:*

*Задание 3:*

Создать окно, реализовать анимацию, с помощью картинки, состоящей из нескольких кадров.

*Решение:*

**8.3 Вывод по работе**

# 9 Практическая работа №9

**9.1 Цель работы**

Цель данной лабораторной работы - изучить понятие интерфейса, научиться создавать интерфейсы в Java и применять их в программах.

**9.2 Теоретическое введение**

Механизм наследования очень удобен, но он имеет свои ограничения. В частности, мы можем наследовать только от одного класса, в отличие, например, от языка С++, где имеется множественное наследование.

В языке Java подобную проблему позволяют решить интерфейсы. Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов.

Чтобы определить интерфейс, используется ключевое слово interface.

Интерфейс может определять различные методы, которые, так же как и абстрактные методы абстрактных классов не имеют реализации. В данном случае объявлен только один метод. Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ public, так как цель интерфейса - определение функционала для реализации его классом. Поэтому весь функционал должен быть открыт для реализации.

И также при объявлении интерфейса надо учитывать, что только один интерфейс в файле может иметь тип доступа public. А его название должно совпадать с именем файла. Остальные интерфейсы (если такие имеются в файле java) не должны иметь модификаторов доступа.

**9.3 Выполнение лабораторной работы**

*Задание 1:*

Создать интерфейс Nameable, с методом getName(), возвращающим имя объекта, реализующего интерфейс. Проверить работу для различных объектов (например, можно создать классы, описывающие разные сущности, которые могут иметь имя: планеты, машины, животные и т. д.).

*Решение:*

Для решения данной задачи был реализован интерфейс Nameable. Его код предоставлен на рисунке 9.1.

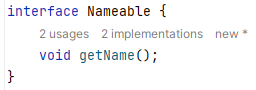


Рисунок 9.1 – Реализация интерфейса Nameable

Также были написаны два класса Planet и Car. Они являются наследниками интерфейса Nameable. Их код реализации предоставлен на рисунке 9.2.

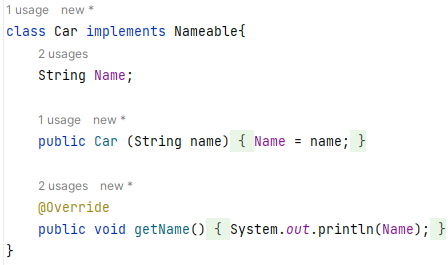
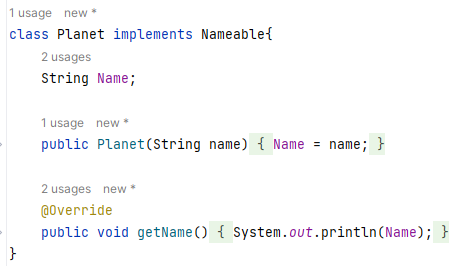


Рисунок 9.2 – Код реализации классов Car и Planet

Результат отработки программы предоставлен на рисунке 9.3.

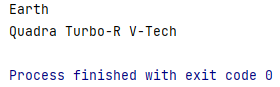


Рисунок 9.3 – Результат отработки программы

*Задание 2:*

Реализовать интерфейс Priceable, имеющий метод getPrice(), возвращающий некоторую цену для объекта. Проверить работу для различных классов, сущности которых могут иметь цену.

*Решение:*

Для решения данной задачи был реализован интерфейс Priceable. Его код предоставлен на рисунке 9.4.

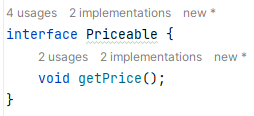


Рисунок 9.4 – Реализация интерфейса Priceable

Также были написаны два класса Clothes и Auto. Они являются наследниками интерфейса Priceable. Их код реализации предоставлен на рисунке 9.5.



Рисунок 9.5 – Код реализации классов Car и Planet

Результат отработки программы предоставлен на рисунке 9.6.

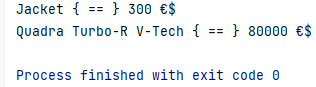


Рисунок 9.6 – Результат отработки программы

**9.3 Вывод по работе**

# 10 Практическая работа №10

**10.1 Цель работы**

Разработка и программирование рекурсивных алгоритмов на языке Java.

**10.2 Теоретическое введение**

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Так же, как и у перебора (цикла) у рекурсии должно быть условие остановки — базовый случай (иначе также, как и цикл, рекурсия будет работать вечно — infinite). Это условие и является тем случаем, к которому рекурсия идет (шаг рекурсии). При каждом шаге вызывается рекурсивная функция до тех пор, пока при следующем вызове не сработает базовое условие и не произойдет остановка рекурсии (а точнее возврат к последнему вызову функции). Всё решение сводится к поиску решения для базового случая. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

Итак, рекурсивная функция состоит из:

1. условие остановки или же базового случая или условия;
2. условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам.

**10.3 Выполнение лабораторной работы**

*Задание 12:*

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите все нечетные числа из этой последовательности, сохраняя их порядок. В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 10.1.

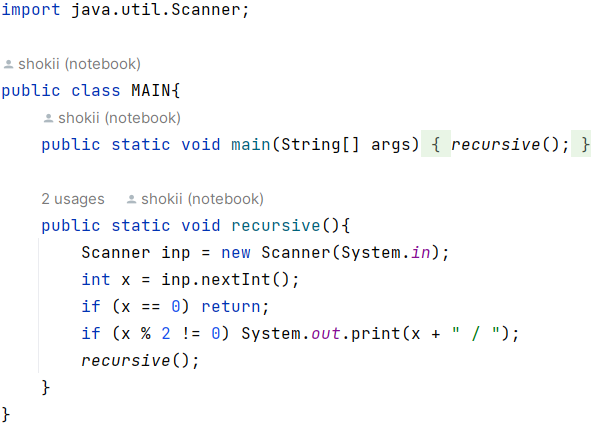


Рисунок 10.1 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 10.2.

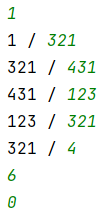


Рисунок 10.2 – Скриншот результата отработки программы

*Задание 13:*

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо. 64 В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 10.3.

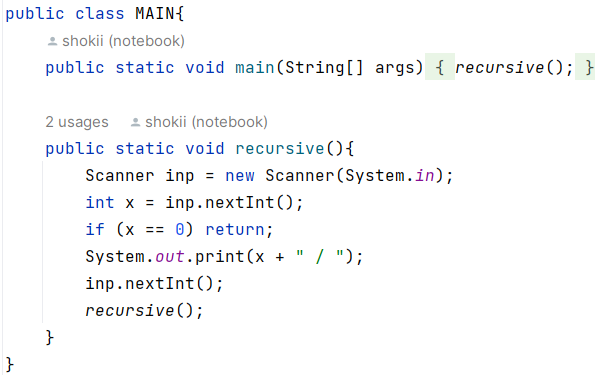


Рисунок 10.3 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 10.4.

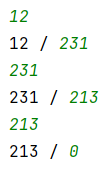


Рисунок 10.4 – Скриншот результата отработки программы

*Задание 14:*

Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 10.5.

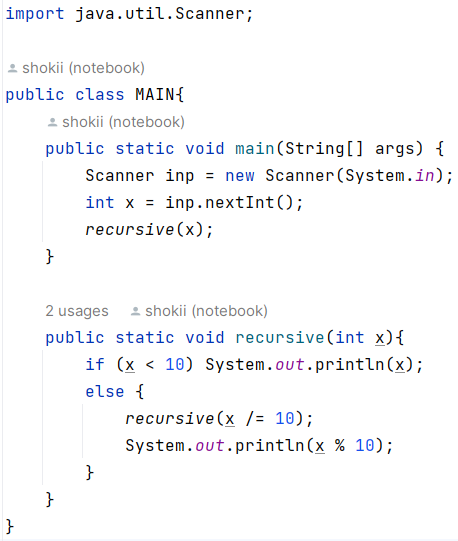


Рисунок 10.5 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 10.6.

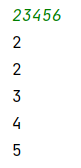


Рисунок 10.6 – Скриншот результата отработки программы

*Задание 15:*

Дано натуральное число N. Выведите все его цифры по одной, в обратном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

*Решение:*

Для решения задачи был создан рекурсивный метод recursive. Который проверяет число на выполнение условия поставленной задачи и, если это так-то выводит его и вызывает функцию recursive. Демонстрация кода предоставлена на рисунке 10.7.

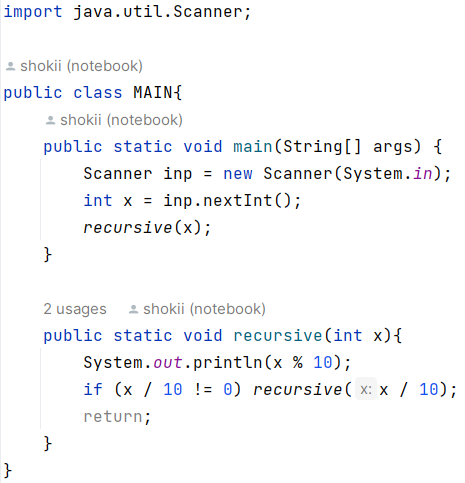


Рисунок 10.7 – Скриншот кода программы

Результат отработки программы вывел данное окно предоставлена на рисунке 10.8.

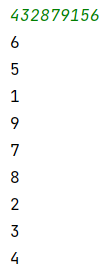


Рисунок 10.8 – Скриншот результата отработки программы

**10.3 Вывод по работе**

В ходе данной практической работы были получены навыки работы с рекурсивными алгоритмами на языке Java.

Рекурсивное программирование помогает оптимизировать программу создать простые решения для трудных задач, используя присущую методу особенность, а именно вызов самого себя. Однако очень важно обеспечить правильное условие для завершения рекурсивного вызов, чтобы избежать бесконечной рекурсии.