|  |  |
| --- | --- |
|  |  |
| 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](#_Toc144647424)

[1.1 Цель работы: 3](#_Toc144647425)

[1.2 Теоретические сведения: 3](#_Toc144647426)

[1.3 Выполнение лабораторной работы: 4](#_Toc144647427)

[1.3 Вывод по работе: 12](#_Toc144647428)

# 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:

• Java Development Kit (JDK);

• Java Runtime Environment (JRE);

• Среда 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 1.1.1) в раздел “File”, затем в раздел “New” и нажать на “Project”

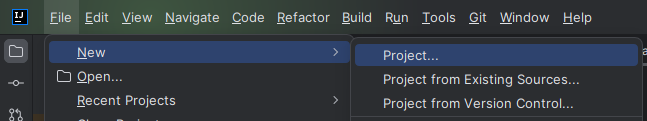


Рисунок 1.1.1- Раздел для создания нового проекта в IntelliJ IDEA

В открывшимся окне (Рисунок 1.1.2), мы можем выбрать название в разделе “Name” и путь по которому у нас будет располагаться проект в разделе “Location”.

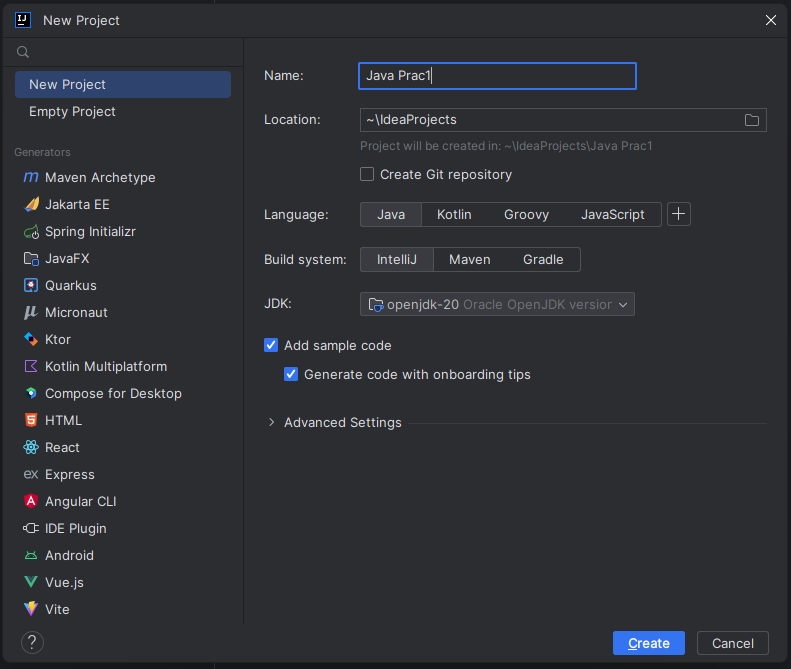


Рисунок 1.1.2 – окно создания нового проекта в IntelliJ IDEA

Написав нужное название и путь для проекта нажимаем кнопку “Create”. Открывшиеся окно (Рисунок 1.1.3) является новый проектом.

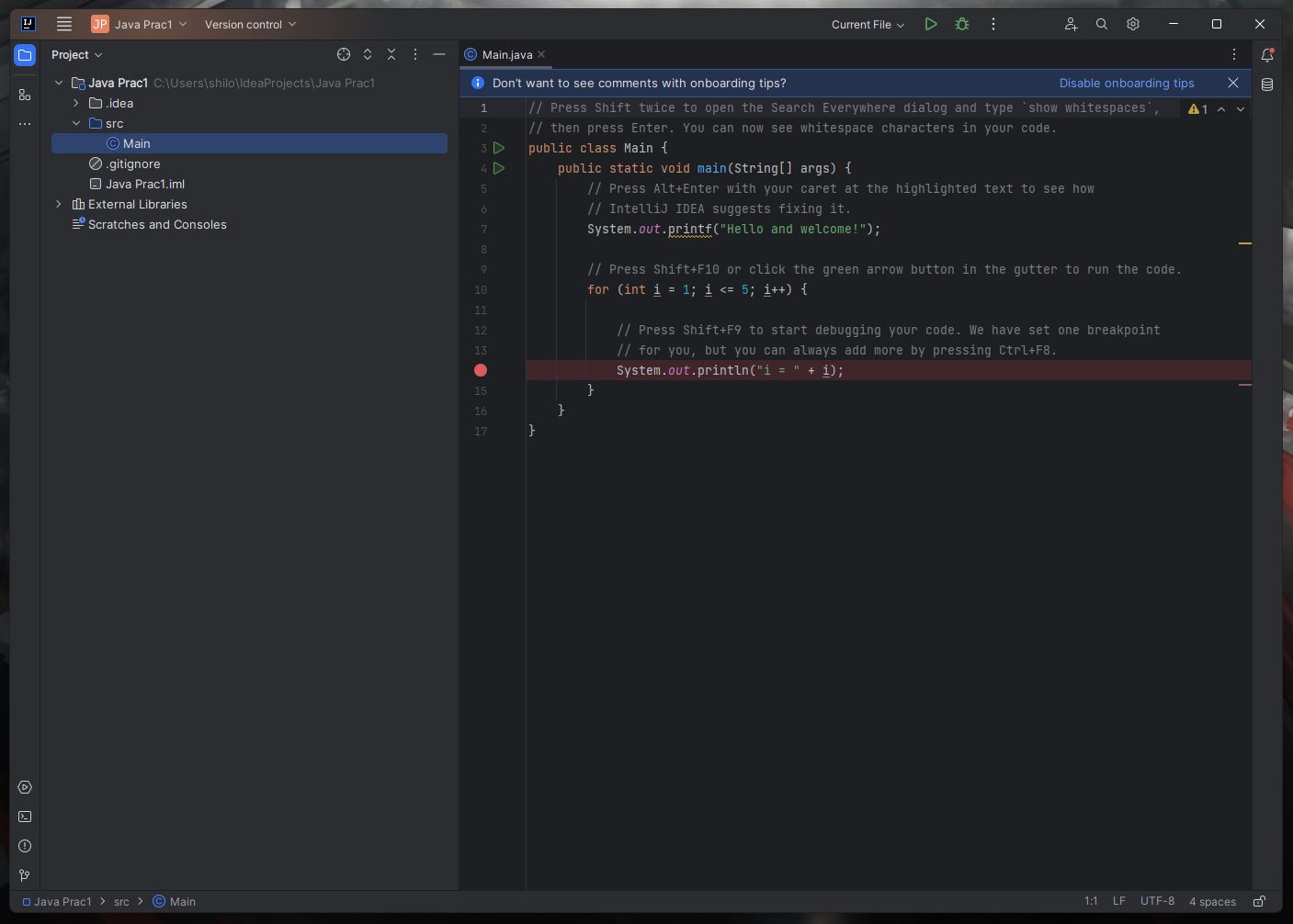


Рисунок 1.1.3 – окно, созданного проекта в IntelliJ IDEA

*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 1.2.1).

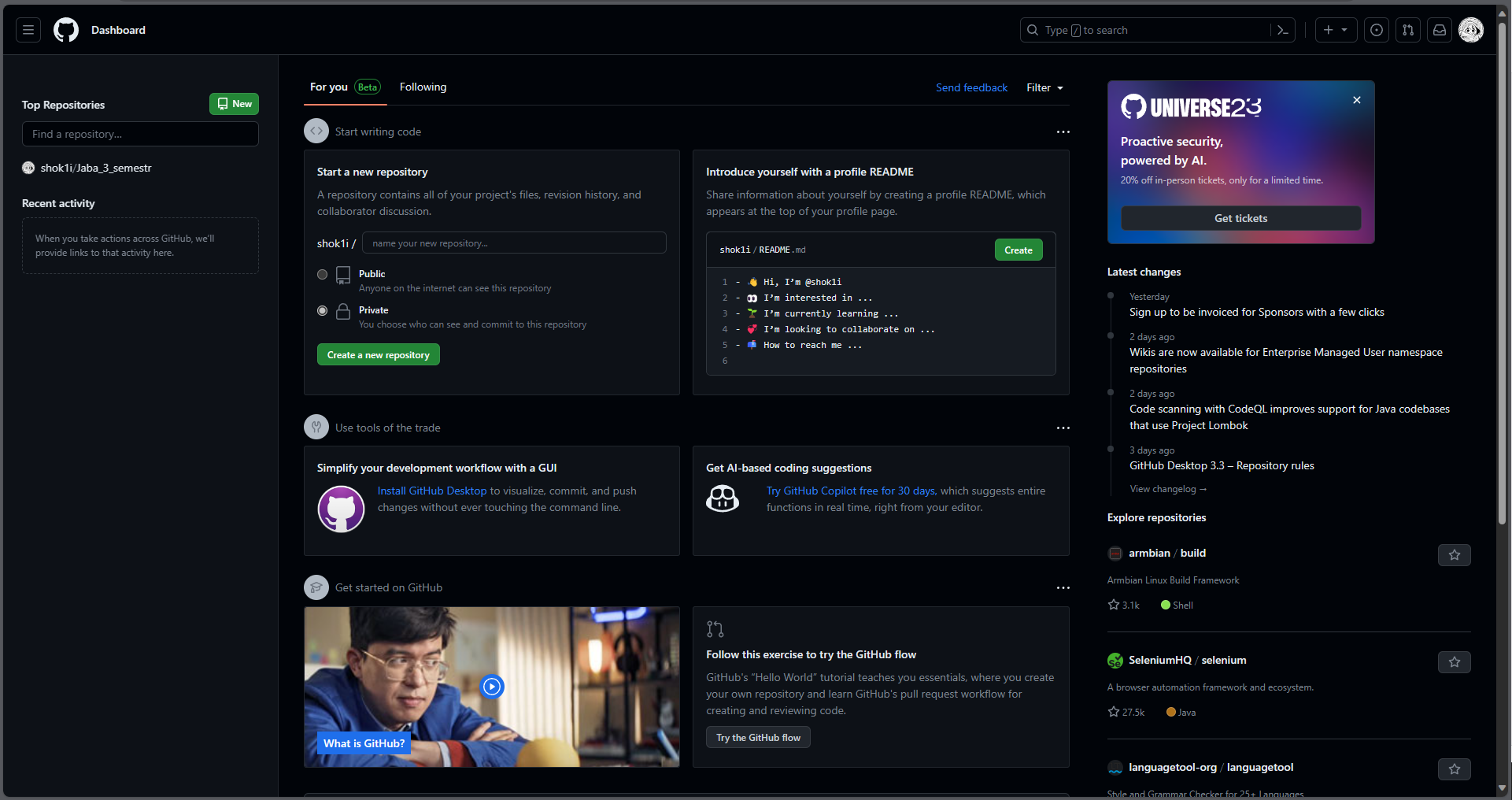


Рисунок 1.2.1 – Главная страница сайта GitHub

Далее требуется ввести название в разделе “Repository name” и нажать “Create repository” (Рисунок 1.2.2).

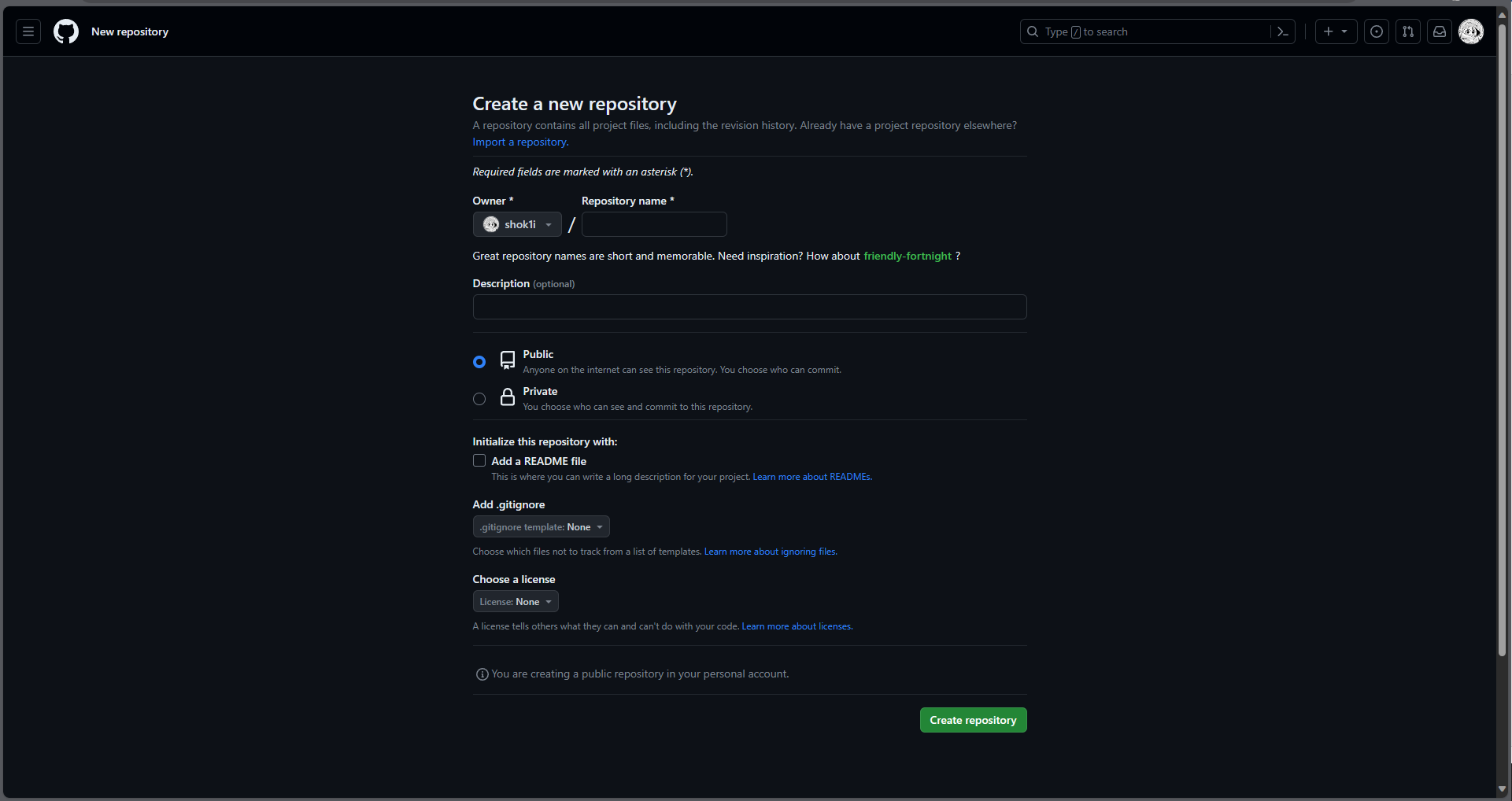


Рисунок 1.2.2 – Страница настройки нового репозитория

Созданный репозиторий предоставлен на рисунке 1.2.3



Рисунок 1.2.3 – 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 1.3.1.

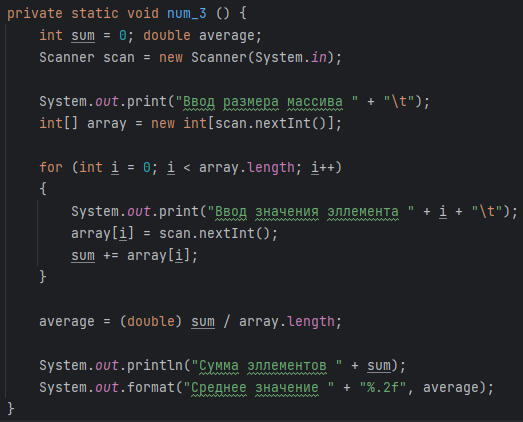


Рисунок 1.3.1 – Код метода num\_3

Демонстрация работы кода предоставлена на рисунке 1.3.2.

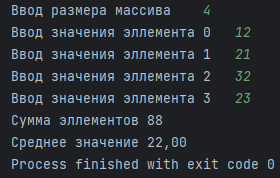


Рисунок 1.3.2 – скриншот работы программы

*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 1.4.1.

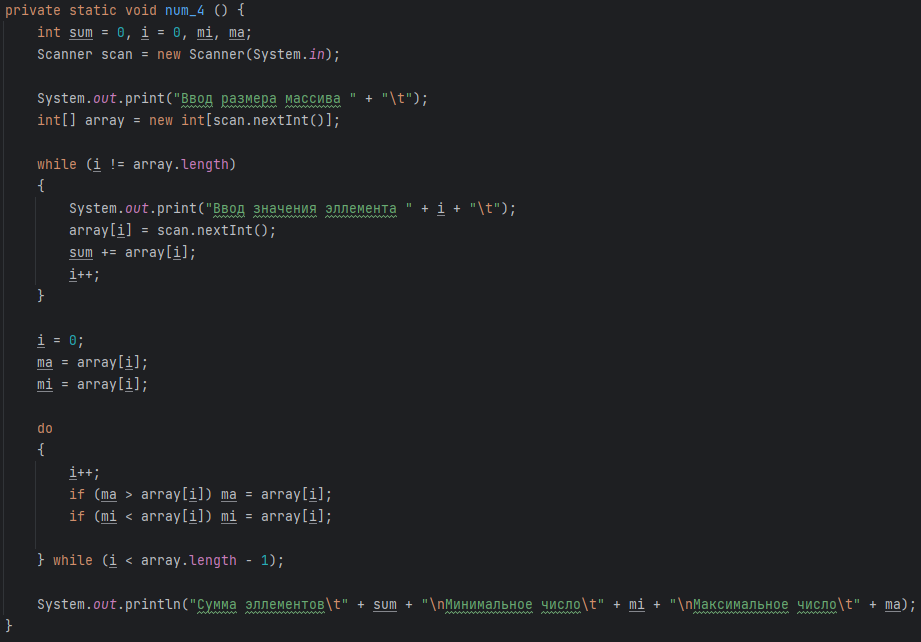


Рисунок 1.4.1. – Код метода num\_4

Демонстрация работы кода предоставлена на рисунке 1.4.2.

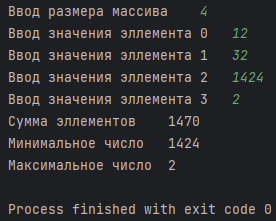


Рисунок 1.4.2 – скриншот работы программы

*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 1.5.1) “Current file” и выбрать “Edit Configuration”

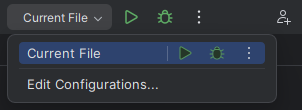


Рисунок 1.5.1 – раздел “Current file”

В открывшимся окне нажимаем на “Add new…” и выбираем “Application” (рисунок 1.5.2)

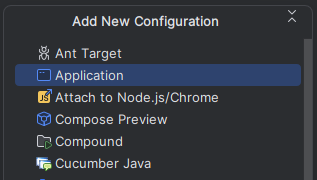


Рисунок 1.5.2 – Скриншот окна с разделами

В открывшимся окне (Рисунок 1.5.3) вписываем в поле “Program arguments” строку, которая будет передано в параметр *args* метода *main*.

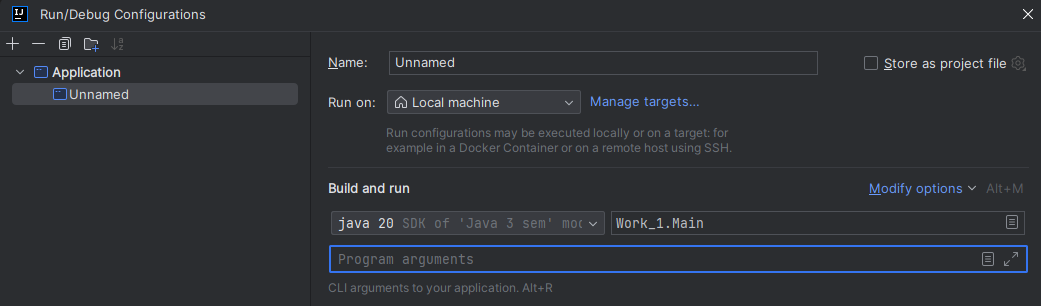


Рисунок 1.5.2 – Скриншот окна в котором можно настроить вводимые параметры при запуске программы

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 1.5.3.



Рисунок 1.5.3 – Код метода num\_5

Демонстрация работы кода предоставлена на рисунке 1.5.4

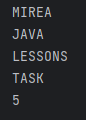


Рисунок 1.5.4 – скриншот работы программы

*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 1.6.1.

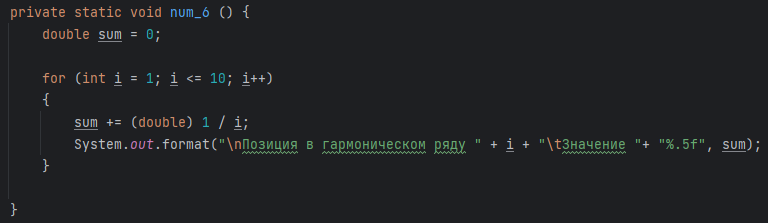


Рисунок 1.6.1. – Код метода num\_6

Демонстрация работы кода предоставлена на рисунке 1.6.2.

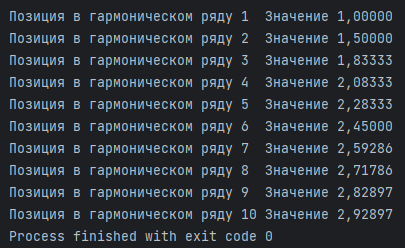


Рисунок 1.6.2 – скриншот работы программы

*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 1.7.1.

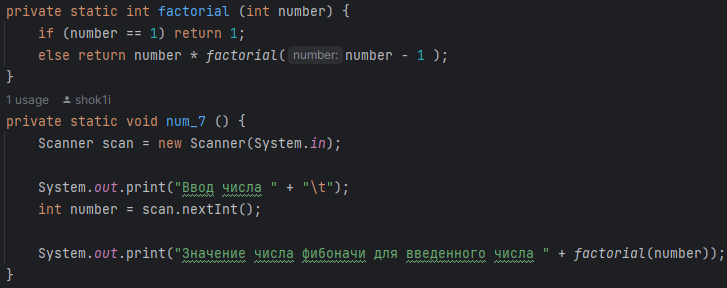


Рисунок 1.7.1. – Код метода num\_7

Демонстрация работы кода предоставлена на рисунке 1.7.2.

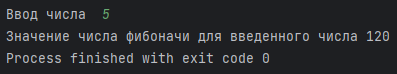


Рисунок 1.7.2 – скриншот работы программы

*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 1.8.1.



Рисунок 1.8.1 – 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:

* private: данные класса доступны только внутри класса;
* protected: данные класса доступны внутри пакета и в наследниках;
* 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.

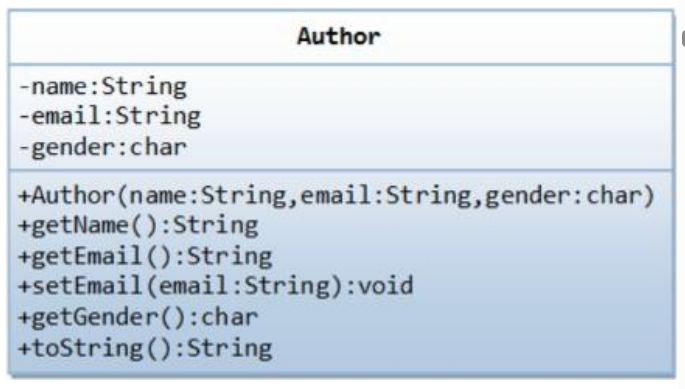


Рисунок 2.1 – Диаграмма класса Author

*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 2.1.1.

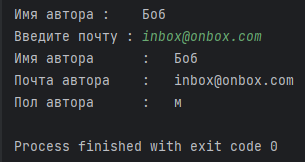


Рисунок 2.1.1 – скриншот работы программы

*Задание 2*

По UML диаграмме класса, представленной на рис. 2.2 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.

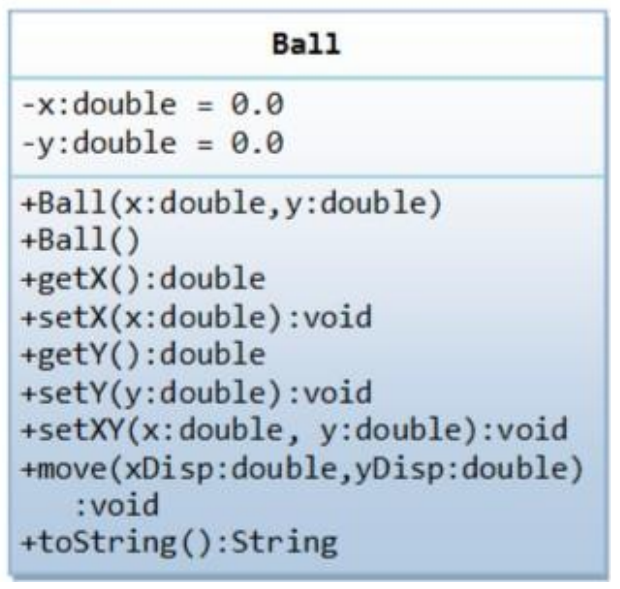


Рисунок 2.2 – Диаграмма класса Ball

*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 Ball*Author* 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 2.2.1.

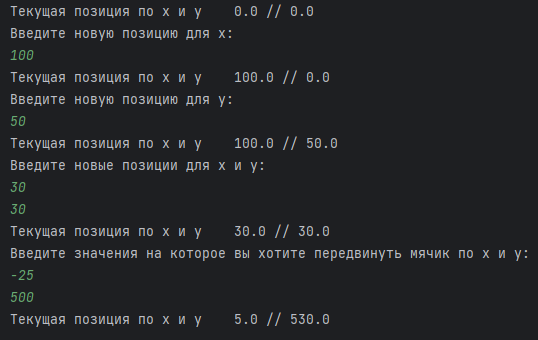


Рисунок 2.1.1 – скриншот работы программы

*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:*

Демонстрация работы кода предоставлена на рисунке 2.3.1.

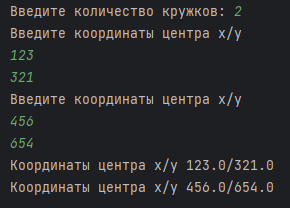


Рисунок 2.3.1 – скриншот работы программы

**2.4 Вывод по работе:**