|  |  |
| --- | --- |
|  |  |
| 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 ИМБО-02-22 Ким К. С.

Принял старший преподаватель *Рачков А.В.*

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

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

Москва 2023

Оглавление

[Практическая работа №1 3](#_Toc146216775)

[Практическая работа №2 10](#_Toc146216776)

[Практическая работа №3 20](#_Toc146216777)

[Практическая работа №5 27](#_Toc146216778)

[Используемая литература 28](#_Toc146216779)

# **Практическая работа №1**

**Цель работы:**

Освоить на практике работу с классами языка программирования Java.

**Теоретическое введение:**

В Java, класс является определением объектов одного и того же вида. Другими словами, класс — это тип данных, создаваемый программистом для решения задач. Он представляет из себя шаблон, или прототип, который определяет и описывает статические свойства и динамическое поведение, общие для всех объектов одного и того же вида.

Экземпляр класса - реализация конкретного объекта типа класс.

Другими словами, экземпляр экземпляра класса. Все экземпляры класса

имеют аналогичные свойства, как задано в определении класса. Например, вы можете определить класс с именем "Студент " и создать три экземпляра класса "Студент": " Петр", " Павел" и " Полина ". Термин "Объект " обычно относится к экземпляру класса. Но он часто используется свободно, которые могут относиться к классу или экземпляру.

**Выполнения лабораторной работы**

*Задание №1:*

Реализуйте простейший класс «Собака»

*Решение*:

Данное решение представлено на Листинге – 1 и Листинге – 2

*Листинг 1 – Dog.java*

|  |
| --- |
| *package dog;*  *import java.lang.\*;*  *public class Dog {*  *private String name;*  *private int age;*  *public Dog(String n, int a) {*  *name = n;*  *age = a;*  *}*  *public Dog(String n) {*  *name = n;*  *age = 0;*  *}*  *public Dog() {*  *name = "Pup";*  *age = 0;*  *}*  *public void setAge(int age) {*  *this.age = age;*  *}*  *public void setName(String name) {*  *this.name = name;*  *}*  *public String getName(String name) {*  *return name;*  *}*  *public int getAge() {*  *return age;*  *}*  *public String toString() {*  *return this.name + ", age " + this.age;*  *}* |

*Продолжение Листинга 1*

|  |
| --- |
| *public void intoHumanAge() {*  *System.out.println(name + "'s age in human years is " + age \* 7 + " years");*  *}*  *}* |

*Листинг 2 – TestDog.java*

|  |
| --- |
| package dog;  import java.lang.\*;  public class TestDog {  public static void main(String[] args) {  Dog d1 = new Dog("Mike", 2);  Dog d2 = new Dog("Helen", 7);  Dog d3 = new Dog("Bob");  d3.setAge(1);  System.out.println(d1);  d1.intoHumanAge();  d2.intoHumanAge();  d3.intoHumanAge();  }  } |

*Результат выполнения программ:*

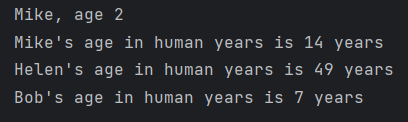


Рисунок 1 – Результат задачи №1

*Задание №2:*

Реализуйте простейший класс «Мяч»

*Решение*:

Данное решение представлено на Листинге – 3 и Листинге – 4

*Листинг 3 – Ball.java*

|  |
| --- |
| package ball;  import java.lang.\*;  public class Ball {  private String name;  private String material;  public Ball(String n, String m) {  name = n;  material = m;  }  public Ball(String n) {  name = n;  }  public Ball() {  name = "Untitled";  material = "Untitled";  }  public void setName(String name) {  this.name = name;  }  public void setmaterial(String material) {  this.material = material;  }  public String getName(String name) {  return name;  }  public String getmaterial(String material) {  return material;  }  @Override  public String toString() {  return this.name + " мяч, Материал: "+ this.material;  }  } |

*Листинг 4 – TestBall.java*

|  |
| --- |
| package ball;  public class TestBall {  public static void main(String[] args) {  Ball b1 = new Ball("Футбольный", "латекс");  System.out.println(b1);  b1.setName("Баскетбольный");  b1.setmaterial("кожа");  System.out.println(b1);  }  } |

*Результат выполнения программ:*

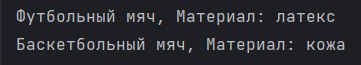


Рисунок 2 – Результат задачи №2

*Задание №3:*

Реализуйте простейший класс «Книга»

*Решение*:

Данное решение представлено на Листинге – 5 и Листинге – 6

*Листинг 5 – Book.java*

|  |
| --- |
| package book;  import java.lang.\*;  public class Book {  private String name;  private String author;  private int year;  public Book(String n, String a, int y) {  name = n;  author = a;  year = y;  }  public Book(String n, String a) {  name = n;  author = a;  year = 0;  }  public Book() {  name = "Untitled";  author = "Untitled";  year = 0;  }  public void setYear(int year) {  this.year = year;  }  public void setName(String name) {  this.name = name;  }  public void setAuthor(String author) {  this.author = author;  }  public String getName(String name) {  return name;  }  public String getAuthor(String author) {  return author;  }  public int getYear() {  return year;  }  @Override  public String toString() {  return "Название книги: " + this.name + ", Автор книги: " + this.author + ", Год выпуска: " + this.year;  }  } |

*Листинг 6 – TestBook.java*

|  |
| --- |
| package book;  public class TestBook {  public static void main(String[] args) {  Book b1 = new Book("Преступление и наказание", "Ф.М.Достоевский", 1866);  Book b2 = new Book("1984", "Джордж Оруэлл", 1949);  Book b3 = new Book("Игра престолов", "Джордж Р.Р. Мартин");  b3.setYear(1996);  System.out.println(b1);  System.out.println(b2);  System.out.println(b3);  }  } |

*Результат выполнения программ:*

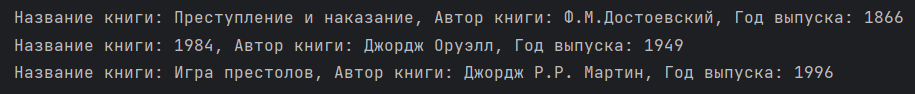


Рисунок 1 – Результат задачи №1

*Выводы по работе:*

В результате выполнения работы научились понимать классы, чем отличается модификатор контроля доступа public и private. Создавать методы классов, конструкторы. в языке Java.

# **Практическая работа №2**

**Цель работы:**

Целями данной работы являются получение практических навыков разработки программ, изучение синтаксиса языка Java, освоение основных конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научиться осуществлять стандартный ввод/вывод данных.

**Теоретическое введение:**

Язык Джава— это объектно-ориентированный язык программирования, с со строгой инкапсуляцией и 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. Например, IDE IntelliJ IDEA или NetBeans

**Выполнения лабораторной работы**

*Задание №1:*

Написать программу, в результате которой массив чисел создается с помощью инициализация (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*:

Данное решение представлено на Листинге – 1

*Листинг 1*

|  |
| --- |
| package ru.mirea.lab1;  import java.util.Scanner;  public class array\_1 {  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  int num = in.nextInt();  double a = 0;  int[] arr = new int[num];  for (int i = 0; i < num; i++) {  arr[i] = in.nextInt();  a += arr[i];  }  System.out.println(a);  double ans = a / num;  System.out.println(ans);  }  } |

*Результат выполнения программ:*

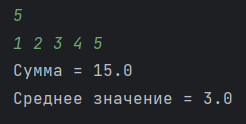
**

Рисунок 1 – Результат задачи №3

*Задание №2:*

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*:

*Листинг 2*

|  |
| --- |
| package ru.mirea.lab1;  import java.util.Scanner;  import java.util.ArrayList;  import java.util.Collections;  public class while\_2 {  public static void main(String[] args) {  System.out.println("Решаем через while");  System.out.println("Введи число");  Scanner in = new Scanner(System.in);  int num = in.nextInt();  if (num > 0) {  var arr = new ArrayList<Integer>(num);  boolean logic = true;  while (logic) {  int i = in.nextInt();  arr.add(i); |

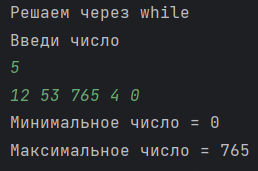
*Продолжение Листинга 2*

|  |
| --- |
| if (arr.size() == num) {  break;  }  }  int min = Collections.min(arr);  int max = Collections.max(arr);  System.out.println(min);  System.out.println(max);  }  else {  System.out.println("Ввел не то число");  }  System.out.println("Решаем через do-while");  System.out.println("Введи число");  int num1 = in.nextInt();  if (num1 > 0) {  var arr1 = new ArrayList<Integer>(num1);  do {  int i = in.nextInt();;  arr1.add(i);  num1--;  } while (num1 != 0);  int min1 = Collections.min(arr1); |

*Продолжение Листинга 2*

|  |
| --- |
| int max1 = Collections.max(arr1);  System.out.println(min1);  System.out.println(max1);  }  else {  System.out.println("Ввел не то число");  }  }  } |

*Результат выполнения программ:*

**

**Рисунок 2 – Результат задачи №4**

*Задание №3:*

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

*Решение*:

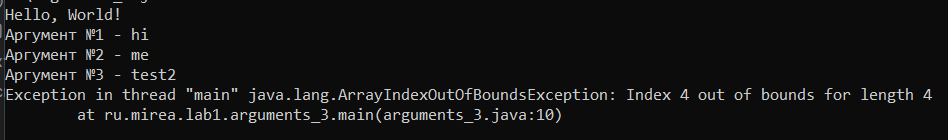
*Листинг 3*

|  |
| --- |
| package ru.mirea.lab1;  public class arguments\_3 {  public static void main(String[] args) {  System.out.println("Hello, World!"); |

*Продолжение Листинга 3*

|  |
| --- |
| for (int i = 1; i <= args.length; i++) {  System.out.println("Аргумент №" + i + " - " + args[i]);  }  }  } |

*Результат выполнения программ:*



**Рисунок 3 – Результат задачи №5**

*Задание №4:*

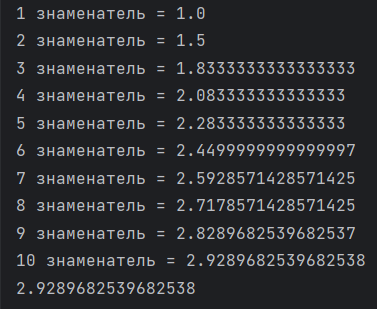
Написать программу, в результате работы которой выводятся на экран первые 10 чисел гармонического ряда (форматировать вывод).

*Решение*:

*Листинг 4*

|  |
| --- |
| package ru.mirea.lab1;  import java.util.Scanner;  public class harmonic\_view\_4  {  public static void main(String[] args)  {  Scanner in = new Scanner(System.in);  int num = 10;  double n = 0;  for (int i = 1; i <= num; i++)  {  n += (double) 1 / i;  System.out.println(i + " знаменатель = " + n);  }  System.out.println(n);  }  } |

*Результат выполнения программ*:



**Рисунок 4 – Результат задачи №6**

*Задание №5:*

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

*Решение*:

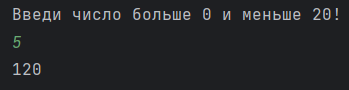
*Листинг 5*

|  |
| --- |
| package ru.mirea.lab1;  import java.util.Scanner;  public class factorial\_5  {  public static void main(String[] args)  {  while (true)  {  System.out.println("Введи число больше 0 и меньше 20!");  Scanner in = new Scanner(System.in);  long num = in.nextInt();  if (num < 0)  {  System.out.println("Введи число больше 0!"); |

*Продолжение Листинга 5*

|  |
| --- |
| }  else if (num > 20)  {  System.out.println("Введи другое меньше 20!");  }  else  {  long prod = 1;  for (int i = 1; i <= num; i++)  {  prod \*= i;  }  System.out.println(prod);  break;  }  }  }  } |

*Результат выполнения программ*:

****

**Рисунок 5 – Результат задачи №7**

*Выводы по работе:*

В результате выполнения работы научились создавать проекты в IDE IntelliJ IDEA и создавать свой собственный Git репозиторий. Узнали, как пользовать переменными, массивами, условными/цикловыми конструкциями и методами в языке Java.

# **Практическая работа №3**

**Цель работы:**

Работа с UML-диаграммами классов

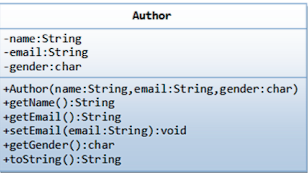
**Теоретическое введение:**

Язык моделирования Unified Modeling Language (UML) является стандартом де-факто с 1998 года для проектирования и документирования объектно-ориентированных программ. Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

**Выполнения 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 2.4.

**

**Рисунок 2.4 – Диаграмма класса Author**

*Решение*:

Данное решение представлено на Листинге – 1 и Листинге – 2

*Листинг 1 – Author.java*

|  |
| --- |
| package mirea.lab2\_1;  import java.lang.\*;  public class Author {  private String name;  private String email;  private char gender;  public Author (String name, String email, char gender) {  this.name = name;  this.email = email;  this.gender = gender;  }  public String getName() {  return name;  } |

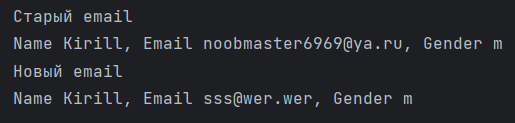
*Продолжение Листинга 1*

|  |
| --- |
| public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public char getGender() {  return gender;  }  @Override  public String toString() {  return "Name " + name + ", Email " + email + ", Gender " + gender;  }  } |

*Листинг 2 – TestAuthor.java*

|  |
| --- |
| package mirea.lab2\_1;  import java.lang.\*;  public class TestAuthor {  public static void main(String[] args) {  Author test = new Author("Kirill", "noobmaster6969@ya.ru", 'm');  test.setEmail("sss@wer.wer");  System.out.println(test.toString());  }  } |

*Результат выполнения программ*:

****

**Рисунок 6 – Результат задачи №1**

*Задание №2:*

По UML диаграмме класса, представленной на рисунке 2.5. написать

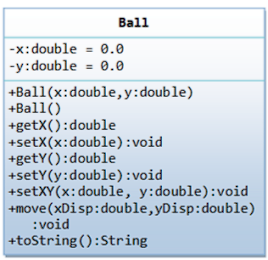
программу, которая состоит из двух классов. Один из них Ball должен

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.

**

**Рисунок 2.5 – Диаграмма класса Ball**

*Решение*:

Данное решение представлено на Листинге – 3 и Листинге – 4

*Листинг 3 – Ball.java*

|  |
| --- |
| package mirea.lab2\_2;  public class Ball {  private double x = 0.0;  private double y = 0.0;  public Ball (double x, double y) {  setXY(x, y);  }  private double x = 0.0;  private double y = 0.0;  public Ball (double x, double y) {  setXY(x, y);  } |

*Продолжение Листинга 3*

|  |
| --- |
| public Ball (double x, double y) {  setXY(x, y);  }  public double getX() {  return x;  }  public void setX(double x) {  this.x = x;  }  public double getY() {  return y;  }  public void setY(double y) {  this.y = y;  }  public void setXY(double x, double y) {  this.x = x;  this.y = y;  }  public void move(double xDisp, double yDisp) {  setXY(x + xDisp, y + yDisp);  }  @Override  public String toString() {  return "Ball: x = " + x + ", y = " + y;  }  } |

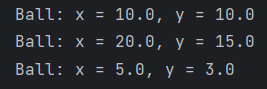
*Листинг 4 – TestBall.java*

|  |
| --- |
| package mirea.lab2\_2;  import java.lang.\*;  public class TestBall {  public static void main(String[] args) {  Ball ball1 = new Ball(10, 10); |

*Продолжение Листинга 4*

|  |
| --- |
| System.out.println(ball1.toString());  ball1.move(10, 5);  System.out.println(ball1.toString());  Ball ball2 = new Ball(5, 3);  System.out.println(ball2.toString());  }  } |

*Результат выполнения программ*:

****

**Рисунок 7 – Результат задачи №2**

*Выводы по работе:*

В результате выполнения научились работать с UML-диаграммами классов. Это удобно представлять графически представлять класс в виде прямоугольника.

# **Практическая работа №5**

**Цель работы:**

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

**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.

# **Используемая литература**

Конспект лекций по дисциплине «Программирование на языке Джава», РТУ МИРЭА, лектор – старший преподаватель Зорина Н.В.