|  |  |
| --- | --- |
|  |  |
| 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](#_Toc145534249)

[Практическая работа №2 13](#_Toc145534250)

[Практическая работа №3 19](#_Toc145534251)

[Практическая работа №4 21](#_Toc145534252)

[Практическая работа №5 22](#_Toc145534253)

[Практическая работа №6 23](#_Toc145534254)

[Практическая работа №7 24](#_Toc145534255)

[Практическая работа №8 25](#_Toc145534256)

[Практическая работа №9 26](#_Toc145534257)

[Практическая работа №10 27](#_Toc145534258)

[Практическая работа №11 28](#_Toc145534259)

[Практическая работа №12 29](#_Toc145534260)

[Практическая работа №13 30](#_Toc145534261)

[Практическая работа №14 31](#_Toc145534262)

[Практическая работа №15 32](#_Toc145534263)

[Практическая работа №16 33](#_Toc145534264)

[Практическая работа №17 34](#_Toc145534265)

[Практическая работа №18 35](#_Toc145534266)

[Практическая работа №19 36](#_Toc145534267)

[Практическая работа №20 37](#_Toc145534268)

[Практическая работа №21 38](#_Toc145534269)

[Практическая работа №22 39](#_Toc145534270)

[Практическая работа №23 40](#_Toc145534271)

[Практическая работа №24 41](#_Toc145534272)

[Используемая литература 42](#_Toc145534273)

# **Практическая работа №1**

**Цель работы:**

Знакомство со средой разработки и написание программ на языке программирования Джава

**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 №3:*

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:*

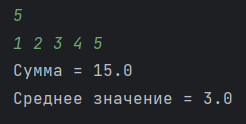
**

Рисунок 1 – Результат задачи №3

*Задание №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*:

*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); |

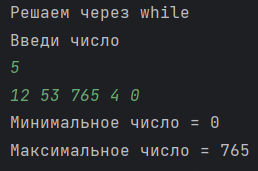
*Продолжение Листинга 2*

|  |
| --- |
| 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 №5:*

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

*Решение*:

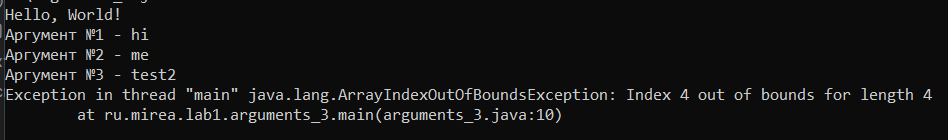
*Листинг 3*

|  |
| --- |
| 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 №6:*

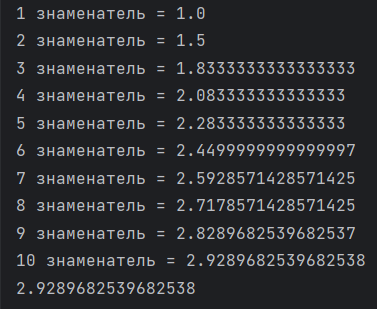
Написать программу, в результате работы которой выводятся на экран первые 10 чисел гармонического ряда (форматировать вывод).

*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 №7:*

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

*Решение*:

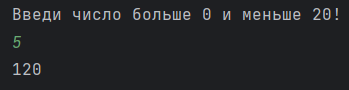
*Листинг 5*

|  |
| --- |
| 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 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 №2**

**Цель работы:**

Работа с UML-диаграммами классов

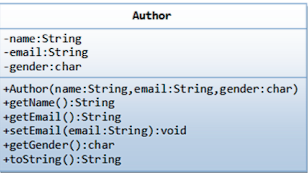
**Теоретическое введение:**

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;  } |

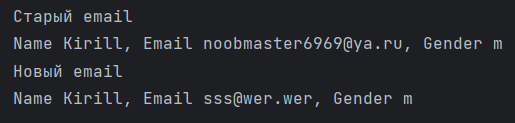
*Продолжение Листинга 1*

|  |
| --- |
| 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

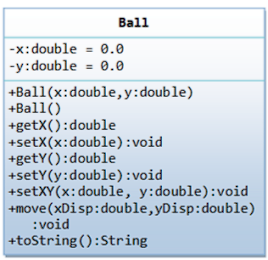
программу, которая состоит из двух классов. Один из них Ball должен

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;  }  } |

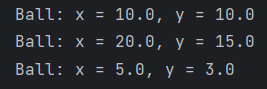
*Листинг 4 – TestBall.java*

|  |
| --- |
| 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 №3**

**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.

# **Практическая работа №4**

**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.

# **Практическая работа №5**

**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.

# **Практическая работа №6**

**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.

# **Практическая работа №7**

**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.

# **Практическая работа №8**

**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.

# **Практическая работа №9**

**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.

# **Практическая работа №10**

**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.

# **Практическая работа №11**

**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.

# **Практическая работа №12**

**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.

# **Практическая работа №13**

**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.

# **Практическая работа №14**

**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.

# **Практическая работа №15**

**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.

# **Практическая работа №16**

**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.

# **Практическая работа №17**

**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.

# **Практическая работа №18**

**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.

# **Практическая работа №19**

**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.

# **Практическая работа №20**

**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.

# **Практическая работа №21**

**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.

# **Практическая работа №22**

**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.

# **Практическая работа №23**

**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.

# **Практическая работа №24**

**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.

# **Используемая литература**

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