|  |  |
| --- | --- |
|  |  |
| 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 ИКБО-02-21 Семянников Н.С.

Принял старший преподаватель *Рачков А.В.*

Практические работы работа выполнены «\_\_\_»\_\_\_\_\_\_\_2022г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2022г.

Москва 2022

Оглавление

[Практическая работа № 1 3](#_Toc115892797)

[Практическая работа № 2 8](#_Toc115892798)

[Практическая работа № 3 11](#_Toc115892799)

[Практическая работа № 4 20](#_Toc115892800)

[Практическая работа № 5 23](#_Toc115892801)

[Практическая работа № 6 25](#_Toc115892802)

[Практическая работа № 7 26](#_Toc115892803)

[Практическая работа № 18 27](#_Toc115892804)

[Практическая работа № 19 29](#_Toc115892805)

[Практическая работа № 20 32](#_Toc115892806)

[Используемая литература 40](#_Toc115892807)

# Практическая работа № 1

## Цель работы

Изучить работу с классами в Java

## Теоретическое введение

Класс 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.

Переменные: содержит статические атрибуты класса

Методы: описывают динамическое поведение класса.

Переменные и методы, входящие в состав класса, формально называется переменные-поля данных класса и методы класса.

Для ссылки на переменную-поле данных класса или метод, вы должны:

* сначала создать экземпляр класса, который вам нужен;
* затем, использовать оператор точка “.” чтобы сослаться на элемент класса.

Конструктор – это специальный метод класса, который используется для создания и инициализации всех переменных-полей данных класса.

Перегрузка методов означает, что несколько методов могут иметь то же самое имя метод, но сами методы могут иметь различные реализации. Тем не менее, различные реализации должны быть различимы по списку их аргументов

Контроль за доступом осуществляется с помощью модификатора, он может быть использован для управления видимостью класса или переменных – полей или методов внутри класса.

* public: класс / переменная / метод доступными для всех других объектов в системе
* private: класс / переменная / метод доступными в пределах только этого класса

## Выполнение лабораторной работы

### Задание:

Необходимо реализовать простейший класс на языке программирования Java. Не забудьте добавить метод toString() к вашему классу. Так-же в программе необходимо предусмотреть класс-тестер для тестирования класса и вывода информации об объекте.

### Решение:

#### Класс «Собака»

*Листинг 1 - Код "Собака"*

|  |
| --- |
| 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 String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String toString() {  return this.name + ", age " + this.age;  }  public void intoHumanAge() {  System.out.println(name + "'s age in human years is " + age \* 7 + " years");  }  } |

#### Класс-тестер «Собака»

*Листинг 2 - Код тестер "Собака"*

|  |
| --- |
| public class TestDog {  public static void main(String[] args) {  Dog d1 = new Dog("Mike", 3);  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 тестирования класса «Собака» представлен на рисунке 1

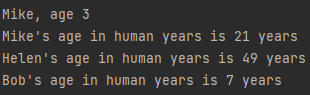


Рисунок 1 - Результат тестирования класса «Собака»

#### Класс «Мяч»

*Листинг 3 - Код "Мяч"*

|  |
| --- |
| public class Circle {  private double radius;  private String color;  public Circle(double r, String cl) {  radius = r;  color = cl;  }  public Circle(double r) {  radius = r;  color = "Red";  }  public Circle() {  radius = 5;  color = "Red";  }  public double getRadius() {  return radius;  }  public void setRadius(double r) {  radius = r;  }  public String getColor() {  return color;  }  public void setColor(String cl) {  color = cl;  }  public String toString() {  return radius + " - " + color;  }  public void getArea() {  System.out.println(Math.PI \* Math.pow(radius, 2));  }  } |

#### Класс-тестер «Мяч»

*Листинг 4 - Код тестер "Мяч"*

|  |
| --- |
| public class TestCircle {  public static void main(String[] args) {  Circle c1 = new Circle(3, "Green");  Circle c2 = new Circle(8);  Circle c3 = new Circle();  c1.setColor("Blue");  System.out.println(c1);  System.out.println(c2);  System.out.println(c3);  c2.getArea();  }  } |

#### Результат тестирования класса «Мяч» представлен на рисунке 2

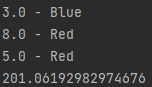


Рисунок 2 - Результат тестирования класса «Мяч»

#### Класс «Книга»

*Листинг 5 - Код "Книга"*

|  |
| --- |
| public class Book {  private int pages;  private String name;  public Book(int p, String n) {  pages = p;  name = n;  }  public Book(int p) {  pages = p;  name = "Lion The Fat";  }  public Book() {  pages = 200;  name = "Lion The Fat";  }  public int getPages() {  return pages;  }  public void setPages(int p) {  pages = p;  }  public String getName() {  return name;  }  public void setName(String n) {  name = n;  }  public String toString() {  return pages + " - " + name;  }  public void getWeight() {  System.out.println(pages \* 1.5 + " grams");  }  } |

#### Класс-тестер «Книга»

*Листинг 6 - Код тестер "Книга"*

|  |
| --- |
| public class TestBook {  public static void main(String[] args) {  Book b1 = new Book(500, "Slim Lion");  Book b2 = new Book(800);  Book b3 = new Book();  b2.setName("Happy Man");  System.out.println(b1);  System.out.println(b2);  System.out.println(b3);  b2.getWeight();  }  } |

#### Результат тестирования класса «Книга» представлен на рисунке 3

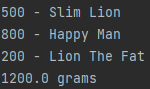


Рисунок 3 - Результат тестирования класса «Книга»

## Выводы по работе:

Изучил инкапсуляцию и работу с классами в Java.

# Практическая работа № 2

## Цель работы

Изучить обозначения и работу с UML диаграммами.

## Теоретическое введение

Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

Графически представляем класс в виде прямоугольника, разделенного на три области – область именования класса, область инкапсуляции данных и область операций.

Имя: определяет класс.

Переменные: содержит статические атрибуты класса, или описывают свойства класса

Методы: описывают динамическое поведение класса.

На рисунке 4 приведен общий вид UML диаграммы класса.



Рисунок 4 – UML

## Выполнение лабораторной работы

### Задание:

Необходимо написать программу, которая состоит из двух классов Author и TestAuthor.

Написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса.

### Решение:

#### Класс «Автор»

*Листинг 7 - Код "Автор"*

|  |
| --- |
| package T1;  public class Author {  private String name, email;  private char gender;  public Author(String name, String email, char gender) {  this.name = name;  this.email = email;  if (gender != 'm' && gender != 'f')  gender = 'u';  this.gender = gender;  }  public String getName() {  return name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public char getGender() {  return gender;  }  public String toString() {  return name + "(" + (gender == 'f' ? "ms" : gender) + ") at " + email;  }  } |

#### Класс-тестер «Автор»

*Листинг 8 - Код тестер "Автор"*

|  |
| --- |
| package T1;  public class TestAuthor {  public static void main(String[] args) {  Author a1 = new Author("Ivan Popov", "ivan@mail.ru", 'm');  System.out.println(a1.getName());  System.out.println(a1.getEmail());  System.out.println(a1.getGender());  a1.setEmail("ivanov@yandex.ru");  System.out.println(a1);  }  } |

#### Результат тестирования класса «Автор» представлен на рисунке 5

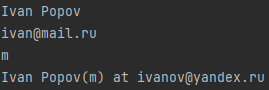


Рисунок 5 - Результат тестирования класса «Автор»

#### Класс «Мяч»

*Листинг 9 - Код "Мяч"*

|  |
| --- |
| package T2;  public class Ball {  private double x = 0.0, y = 0.0;  public Ball(double x, double y) {  this.x = x;  this.y = y;  }  public Ball() {  x = 0;  y = 0;  }  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) {  x += xDisp;  y += yDisp;  }  public String toString() {  return "Lab2.T2.Ball @ (" + x + ", " + y + ")";  }  } |

#### Класс-тестер «Мяч»

*Листинг 10 - Код тестер "Мяч"*

|  |
| --- |
| package T2;  public class TestBall {  public static void main(String[] args) {  Ball b = new Ball(3,0);  System.out.println(b);  b.move(10, 7);  System.out.println(b);  }  } |

#### Результат тестирования класса «Мяч» представлен на рисунке 6



Рисунок 6 - Результат тестирования класса «Мяч»

## Выводы по работе:

Научился писать классы по UML диаграммам.

# 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.

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.

## Выполнение лабораторной работы

### Задание:

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

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

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.

Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable.

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

### Решение:

#### Класс «Shape»

*Листинг 11 - Код "Shape"*

|  |
| --- |
| package T1\_2;  public abstract class Shape {  protected String color;  protected boolean filled;  public Shape() {  color = "Red";  filled = true;  }  public String getColor() {  return color;  }  public void setColor(String color) {  this.color = color;  }  public boolean isFilled() {  return filled;  }  public void setFilled(boolean filled) {  this.filled = filled;  }  public abstract double getArea();  public abstract double getPerimeter();  public abstract String toString();  } |

#### Класс «Circle»

*Листинг 12 - Код "Circle"*

|  |
| --- |
| package T1\_2;  public class Circle extends Shape {  protected double radius;  public Circle() {  radius = 5;  }  public Circle(double r) {  radius = r;  }  public Circle(double r, String cl) {  radius = r;  color = cl;  }  public Circle(double r, String cl, boolean fl) {  radius = r;  color = cl;  filled = fl;  }  public double getRadius() {  return radius;  }  public void setRadius(int radius) {  this.radius = radius;  }  @Override  public double getArea() {  return Math.PI \* radius \* radius;  }  @Override  public double getPerimeter() {  return 2 \* Math.PI \* radius;  }  @Override  public String toString() {  return "Circle {" +  "radius=" + radius +  ", color='" + color + '\'' +  ", filled=" + filled +  '}';  }  } |

#### Класс «Rectangle»

*Листинг 13 - Код "Rectangle"*

|  |
| --- |
| package T1\_2;  public class Rectangle extends Shape {  protected double width, height;  public Rectangle() {  width = 10;  height = 10;  }  public Rectangle(double w) {  width = w;  height = 10;  }  public Rectangle(double w, double h) {  width = w;  height = h;  }  public Rectangle(double w, double h, String cl) {  width = w;  height = h;  color = cl;  }  public Rectangle(double w, double h, String cl, boolean fl) {  width = w;  height = h;  color = cl;  filled = fl;  }  public double getHeight() {  return height;  }  public double getWidth() {  return width;  }  public void setHeight(double height) {  this.height = height;  }  public void setWidth(double width) {  this.width = width;  }  @Override  public double getPerimeter() {  return 2 \* (width + height);  }  @Override  public double getArea() {  return width \* height;  }  @Override  public String toString() {  return "Rectangle {" +  "width=" + width +  ", height=" + height +  ", color='" + color + '\'' +  ", filled=" + filled +  '}';  }  } |

#### Класс «Square»

Листинг 14 - Код "Square"

|  |
| --- |
| package T1\_2;  public class Square extends Rectangle {  public Square() {  width = 5;  height = 5;  }  public Square(double sd) {  width = sd;  height = sd;  }  public Square(double sd, String cl) {  color = cl;  width = sd;  height = sd;  }  public Square(double sd, String cl, boolean fl) {  color = cl;  filled = fl;  width = sd;  height = sd;  }  public double getSide() {  return width;  }  public void setSide(double sd) {  width = sd;  height = sd;  }  @Override  public void setWidth(double width) {  super.width = width;  }  @Override  public void setHeight(double height) {  super.height = height;  }  @Override  public String toString() {  return "Square {" +  ", width=" + width +  ", height=" + height +  ", color='" + color + '\'' +  ", filled=" + filled +  '}';  }  } |

#### Класс «TestAbstract»

*Листинг 15 - Код "TestAbstract"*

|  |
| --- |
| package T1\_2;  public class TestAbstract {  public static void main(String[] args) {  Circle c = new Circle();  Rectangle re = new Rectangle();  Square sq = new Square();  System.out.println(c);  System.out.println("Area: " + c.getArea());  System.out.println("Perimeter: " + c.getPerimeter());  System.out.println("Radius: " + c.getRadius());  System.out.println(re);  System.out.println("Area: " + re.getArea());  System.out.println("Perimeter: " + re.getPerimeter());  System.out.println("Width: " + re.getWidth());  System.out.println(sq);  System.out.println("Area: " + sq.getArea());  System.out.println("Perimeter: " + sq.getPerimeter());  System.out.println("Side: " + sq.getSide());  }  } |

#### Класс «Task3»

*Листинг 16 - Код "Task3"*

|  |
| --- |
| package T3;  import T1\_2.Circle;  import T1\_2.Rectangle;  import T1\_2.Shape;  import T1\_2.Square;  public class Task\_3 {  public static void main(String[] args) {  //Преобразование подкласса в родительский класс  Shape s1 = new Circle(5.5, "RED", false);  System.out.println(s1);  System.out.println(s1.getArea());  System.out.println(s1.getPerimeter());  System.out.println(s1.getColor());  System.out.println(s1.isFilled());  //Нет доступа к методу внутри класса Circle, так как создан объект класса Shape  //System.out.println(s1.getRadius());  //Создаём новый объект класса Circle и, меняя класс у переменной s1, присваиваем её новому объекту  Circle c1 = (Circle)s1;  System.out.println(c1);  System.out.println(c1.getArea());  System.out.println(c1.getPerimeter());  System.out.println(c1.getColor());  System.out.println(c1.isFilled());  System.out.println(c1.getRadius()); //Теперь радиус выводится  //Shape s2 = new Shape(); //Невозможно создать абстрактный класс  //Преобразование подкласса в родительский класс  Shape s3 = new Rectangle(1.0, 2.0, "RED", false);  System.out.println(s3);  System.out.println(s3.getArea());  System.out.println(s3.getPerimeter());  System.out.println(s3.getColor());  //System.out.println(s3.getHeight()); //Нет доступа к методу подкласса  Rectangle r1 = (Rectangle)s3; //Преобразуем родительский класс в подкласс  System.out.println(r1);  System.out.println(r1.getArea());  System.out.println(r1.getColor());  System.out.println(r1.getHeight()); //Появился доступ к методу подкласса  Shape s4 = new Square(6.6); //Преобразование под-подкласса в родительский класс  System.out.println(s4);  System.out.println(s4.getArea());  System.out.println(s4.getColor());  //System.out.println(s4.getSide()); //Нет доступа к методу подкласса  Rectangle r2 = (Rectangle)s4; //Преобразуем родительский класс в подкласс  System.out.println(r2);  System.out.println(r2.getArea());  System.out.println(r2.getColor());  //System.out.println(r2.getSide()); //Доступа к методу класса Square до сих пор нет!  System.out.println(r2.getHeight());  Square sq1 = (Square)r2; //Преобразуем класса Rectangle в Square  System.out.println(sq1);  System.out.println(sq1.getArea());  System.out.println(sq1.getColor());  System.out.println(sq1.getSide()); //Появился доступ к методу подкласса  System.out.println(sq1.getHeight()); //Остался доступ к методу род. класса  }  } |

#### Класс «Movable»

|  |
| --- |
| package T4;  public interface Movable {  public void moveUp();  public void moveDown();  public void moveLeft();  public void moveRight();  } |

#### Класс «MovableCircle»

*Листинг 17 - Код "MovableCircle"*

|  |
| --- |
| package T4;  public class MovableCircle implements Movable {  private int radius;  private MovablePoint center;  public MovableCircle(int x, int y, int xSpeed, int ySpeed) {  center = new MovablePoint(x, y, xSpeed, ySpeed);  }  @Override  public String toString() {  return "MovableCircle {" +  "radius=" + radius +  ", center=" + center +  '}';  }  @Override  public void moveUp() {  center.y += center.ySpeed;  }  @Override  public void moveDown() {  center.y -= center.ySpeed;  }  @Override  public void moveLeft() {  center.x -= center.xSpeed;  }  @Override  public void moveRight() {  center.x += center.xSpeed;  }  } |

#### Класс «MovablePoint»

*Листинг 18 - Код "MovablePoint"*

|  |
| --- |
| package T4;  public class MovablePoint implements Movable {  protected int x, y, xSpeed, ySpeed;  public MovablePoint(int x, int y, int xSpeed, int ySpeed) {  this.x = x;  this.y = y;  this.xSpeed = xSpeed;  this.ySpeed = ySpeed;  }  @Override  public void moveUp() {  y += ySpeed;  }  @Override  public void moveDown() {  y -= ySpeed;  }  @Override  public void moveLeft() {  x -= xSpeed;  }  @Override  public void moveRight() {  x += xSpeed;  }  @Override  public String toString() {  return "MovablePoint {" +  "x=" + x +  ", y=" + y +  ", xSpeed=" + xSpeed +  ", ySpeed=" + ySpeed +  '}';  }  } |

#### Класс «TestMovable»

*Листинг 19 - Код "TestMovable"*

|  |
| --- |
| package T4;  public class TestMovable {  public static void main(String[] args) {  MovablePoint mv = new MovablePoint(10, 0, 3, 7);  MovableCircle mc = new MovableCircle(15, 20, 10, 40);  System.out.println(mv.toString());  System.out.println(mc.toString());  mc.moveUp();  mc.moveRight();  System.out.println(mc);  }  } |

#### Класс «MovableRectangle»

*Листинг 20 - Код "MovableRectangle"*

|  |
| --- |
| package T5;  import T4.Movable;  import T4.MovablePoint;  public class MovableRectangle implements Movable {  private MovablePoint tl, br;  public MovableRectangle(int x1, int y1, int x2, int y2, int xSpeed, int ySpeed) {  tl = new MovablePoint(x1, y1, xSpeed, ySpeed);  br = new MovablePoint(x2, y2, xSpeed, ySpeed);  }  @Override  public void moveUp() {  tl.moveUp();  br.moveUp();  }  @Override  public void moveDown() {  tl.moveDown();  br.moveDown();  }  @Override  public void moveLeft() {  tl.moveLeft();  br.moveLeft();  }  @Override  public void moveRight() {  tl.moveRight();  br.moveRight();  }  @Override  public String toString() {  return "MovableRectangle {" +  "tl=" + tl +  ", br=" + br +  '}';  }  } |

#### Класс «TestRect»

*Листинг 21 - Код "TestRect"*

|  |
| --- |
| package T5;  public class TestRect {  public static void main(String[] args) {  MovableRectangle mr = new MovableRectangle(5, 10, 10, 5, 10, 10);  System.out.println(mr);  mr.moveUp();  mr.moveRight();  System.out.println(mr);  }  } |

## Выводы по работе:

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

# Практическая работа № 4

## Цель работы

Введение в событийное программирование на языке Java.

## Теоретическое введение

TextField - текстовое поле или поля для ввода текста (можно ввести только одну строку).

Компонент TextArea похож на TextField, но в него можно вводить более одной строки.

Мы можем легко добавить возможность прокрутки к текстовому полю, добавив его в контейнер с именем JScrollPane.

### Менеджер BorderLayout:

Разделяет компонент на пять областей (WEST, EAST, NOTH, SOUTH and Center). Другие компоненты могут быть добавлены в любой из этих компонентов пятерками.

### Менеджер GridLayout:

С помощью менеджера GridLayout компонент может принимать форму таблицы, где можно задать число строк и столбцов.

### Менеджер Null Layout Manager:

Иногда бывает нужно изменить размер и расположение компонента в контейнере. Таким образом, мы должны указать программе не использовать никакой менеджер компоновки, то есть (setLayout (нуль)).

### Слушатель событий мыши MouseListener.

Мы можем реализовывать слушателей мыши и также слушателей клавиатуры на компонентах GUI. Интерфейс MouseListener имеет следующие методы:

* mouseClicked
* mouseEntered
* mouseExited
* mousePressed
* mouseReleased

## Выполнение лабораторной работы

### Задание:

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид.

### Решение:

#### Класс «MyMouse»

*Листинг 22 - Код "MyMouse"*

|  |
| --- |
| import com.sun.tools.jconsole.JConsoleContext;  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  import javax.swing.border.Border;  class MyMouse extends JFrame  {  JButton AC = new JButton("AC Milan");  JButton Real = new JButton("Real Madrid");  JLabel Res=new JLabel("Result: 0 X 0");  JLabel Last=new JLabel("Last Scorer: N/A");  JLabel Winner=new JLabel("Winner: DRAW");  public int AC\_int=0;  public int Real\_int=0;  Border border = BorderFactory.createLineBorder(Color.BLACK, 4);  public class TestActionListener implements ActionListener  {  public void actionPerformed(ActionEvent e)  {  AC\_int++;  Res.setText("Result: "+AC\_int+" X "+Real\_int);  Last.setText("Last Scorer: AC Milan");  Winner.setText("Winner: "+((AC\_int>Real\_int)?"AC Milan":((AC\_int==Real\_int)?"DRAW":"Real Madrid")));  }  }  public class TestActionListener1 implements ActionListener  {  public void actionPerformed(ActionEvent e)  {  Real\_int++;  Res.setText("Result: "+AC\_int+" X "+Real\_int);  Last.setText("Last Scorer: Real Madrid");  Winner.setText("Winner: "+((AC\_int>Real\_int)?"AC Milan":((AC\_int==Real\_int)?"DRAW":"Real Madrid")));  }  }  public MyMouse()  {  setLayout(null);  setSize(400,400);  AC.setBounds(50,80,400,60);  ActionListener actionListener = new TestActionListener();  AC.addActionListener(actionListener);  add(AC);  Real.setBounds(460,80,400,60);  ActionListener actionListener1 = new TestActionListener1();  Real.addActionListener(actionListener1);  add(Real);  Res.setBounds(50,30,810,50);  Res.setBorder(border);  Res.setFont(new Font("Serif", Font.PLAIN, 18));  Res.setHorizontalAlignment(JLabel.CENTER);  add(Res);  Last.setBounds(50,150,400,50);  Last.setBorder(border);  Last.setFont(new Font("Serif", Font.PLAIN, 18));  Last.setHorizontalAlignment(JLabel.CENTER);  add(Last);  Winner.setBounds(460,150,400,50);  Winner.setBorder(border);  Winner.setFont(new Font("Serif", Font.PLAIN, 18));  Winner.setHorizontalAlignment(JLabel.CENTER);  add(Winner);  setSize(930, 400);  setLocationRelativeTo(null);  }  public static void main(String[]args)  {  new MyMouse().setVisible(true);  }  } |

## Выводы по работе:

Изучил работу с GUI и создание GUI элементов.

# Практическая работа № 5

## Цель работы

Разработка и программирование рекурсивных алгоритмов на языке Java.

## 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.

Итак, рекурсивная функция состоит из:

• условие остановки или же базового случая или условия;

• условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам.

## Выполнение лабораторной работы

### Задание:

7. Разложение на множители

Дано натуральное число n>1. Выведите все простые множители этого числа в порядке не убывания с учетом кратности. Алгоритм должен иметь сложность O(logn)

8. Палиндром

Дано слово, состоящее только из строчных латинских букв. Проверьте, является ли это слово палиндромом. Выведите YES или NO. При решении этой задачи нельзя пользоваться циклами, в решениях на питоне нельзя использовать срезы с шагом, отличным от 1.

10. Разворот числа

Дано число n, десятичная запись которого не содержит нулей. Получите число, записанное теми же цифрами, но в противоположном порядке. При решении этой задачи нельзя использовать циклы, строки, списки, массивы, разрешается только рекурсия и целочисленная арифметика. Функция должна возвращать целое число, являющееся результатом работы программы, выводить число по одной цифре нельзя.

### Решение:

#### Класс «PrimeFactors»

*Листинг 23 - Код "PrimeFactors"*

|  |
| --- |
| import java.util.Scanner;  public class PrimeFactors {  public static int GetPrimes(int n, int i) {  try {  if (n % i == 0) {  if (isPrime(i, 2)) {  System.out.println(i);  }  }  if (i <= n) {  return GetPrimes(n, i + 1);  } else {  return 1;  }  } catch (java.lang.StackOverflowError e) {  return 0;  }  }  public static boolean isPrime(int n, int i) {  if (n <= 2)  return (n == 2);  if (n % i == 0)  return false;  if (i \* i > n)  return true;  return isPrime(n, i + 1);  }  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  System.out.print("Введите число (n > 2): ");  n = sc.nextInt();  if (n > 1) {  System.out.println("Простые числа: ");  if (GetPrimes(n, 2) == 0) {  System.out.println("Стак переполнился");  }  } else {  System.out.println("Число меньше 2");  }  }  } |

Класс «Palindrome»

*Листинг 24 - Код "Palindrome"*

|  |
| --- |
| import java.util.Scanner;  public class Palindrome {  public static boolean isPalindrome(String s, int len, int i) {  if (i >= len / 2)  return true;  if (s.charAt(i) != s.charAt(len - i - 1))  return false;  return isPalindrome(s, len, i + 1);  }  public static void main(String[] args) {  String s;  Scanner sc = new Scanner(System.in);  System.out.print("Введите строку: ");  s = sc.next();  if (isPalindrome(s.toLowerCase(), s.length(), 0)) {  System.out.println("YES");  } else {  System.out.println("NO");  }  }  } |

Класс «MirrorNumber»

*Листинг 25 - Код "MirrorNumber"*

|  |
| --- |
| import java.util.Scanner;  public class MirrorNumber {  public static int mirrorNumb(int n, int a) {  if (n == 0) {  return a;  }  return mirrorNumb(n / 10, a \* 10 + n % 10);  }  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  n = mirrorNumb(n, 0);  System.out.println(n);  }  } |

## Выводы по работе:

Изучил работу рекурсии на языке Java.

# Практическая работа № 6

## Цель работы

## Теоретическое введение

## Выполнение лабораторной работы

### Задание:

### Решение:

## Выводы по работе:

# Практическая работа № 7

## Цель работы

## Теоретическое введение

## Выполнение лабораторной работы

### Задание:

### Решение:

## Выводы по работе:

# Практическая работа № 18

## Цель работы

Целью работы является изучение работы циклов, ввода/вывода, создание массивов и методов в Java.

## Теоретическое введение

Язык Java — это объектно-ориентированный язык программирования.

Программы написанные на Java могут выполняться на различных операционных системах при наличии необходимого ПО - Java Runtime Environment.

Чтобы объявить переменную, необходимо указать тип переменной и её имя.

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Условие — это конструкция, позволяющая выполнять то или другое действие, в зависимости от логического значения, указанного в условии.

Цикл — это конструкция, позволяющая выполнять определенную часть кода несколько раз. В Java есть три типа циклов for, while, do while.

Для ввода данных используется класс Scanner из библиотеки пакетов.

Методы позволяют выполнять блок кода, из любого другого места, где это доступно. Методы определяются внутри классов.

Методы могут быть статическими (можно выполнять без создания экземпляра класса), не статическими (не могут выполняться без создания экземпляра класса). Методы могут быть открытыми(public), закрытыми(private). Закрытые методы могут вызываться только внутри того класса, в котором они определены. Открытые методы можно вызывать для объекта внутри других классов.

## Выполнение лабораторной работы

### Задание:

Сгенерировать массив целых чисел случайным образом, вывести его на экран, отсортировать его, и снова вывести на экран. (Вариант 4)

### Решение:

#### Класс «RandomArray»

*Листинг 26 - Код "RandomArray"*

|  |
| --- |
| import java.util.Arrays;  import java.util.Random;  import java.util.Scanner;  public class RandomArray {  public static void main(String[] args) {  int n;  System.out.print("Enter array size: ");  Scanner sc = new Scanner(System.in);  n = sc.nextInt();  Random rn = new Random();  int[] a = new int[n];  System.out.println("Original");  for (int i = 0; i < n; i++) {  a[i] = rn.nextInt(1, 30);  System.out.print(a[i]+" ");  }  System.out.println();  Arrays.sort(a);  System.out.println("Sorted");  for (int i = 0; i < n; i++) {  System.out.print(a[i]+" ");  }  }  } |

#### Результат работы класса «RandomArray» представлен на рисунке 7.

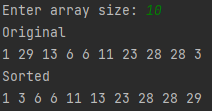


Рисунок 7 - Результат работы класса «RandomArray»

## Выводы по работе:

Изучил работу циклов и создание массива случайных чисел в Java.

# Практическая работа № 19

## Цель работы

Научиться создавать классы.

## Теоретическое введение

Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные.

Объект — это экземпляр класса.

Данные объекта скрыты от остальной программы.

Сокрытие данных называется инкапсуляцией.

Полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Для того чтобы создать класс в языке Java необходимо создать файл с расширением java. Имя файла должно быть таким же, как и имя создаваемого класса.

В теле класса можно описать методы, переменные, константы, конструкторы класса.

Для того чтобы создать экземпляр класса необходимо объявить переменную, тип которой соответствует имени класса или суперкласса. После чего нужно присвоить этой переменной значение, вызвав конструктор создаваемого класса с помощью оператора new.

## Выполнение лабораторной работы

### Задание:

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).

### Решение:

#### Класс «Book»

*Листинг 27 - Код "Book"*

|  |
| --- |
| package V3;  public class Book {  private String author, name;  private int year = 1999, pages = 100;  public Book(String author, String name, int year, int pages) {  this.author = author;  this.name = name;  this.year = year;  this.pages = pages;  }  public Book(String author, String name, int year) {  this.author = author;  this.name = name;  this.year = year;  }  public Book(String author, String name) {  this.author = author;  this.name = name;  }  public String getName() {  return name;  }  public int getPages() {  return pages;  }  public int getYear() {  return year;  }  public String getAuthor() {  return author;  }  public void setAuthor(String author) {  this.author = author;  }  public void setName(String name) {  this.name = name;  }  public void setPages(int pages) {  this.pages = pages;  }  public void setYear(int year) {  this.year = year;  }  @Override  public String toString() {  return "Book {" +  "author='" + author + '\'' +  ", name='" + name + '\'' +  ", year=" + year +  ", pages=" + pages +  '}';  }  } |

#### Класс-тестер «TestBook»

*Листинг 28 - Код тестер "TestBook"*

|  |
| --- |
| package V3;  public class TestBook {  public static void main(String[] args) {  Book b1 = new Book("John Smith", "Atlantis", 2005, 520);  Book b2 = new Book("John Smith", "Atlantis");  System.out.println("Pages: " + b2.getPages());  b2.setName("Capital");  System.out.println(b2 + "\n" + b1);  }  } |

#### Результат тестирования класса «Book» представлен на рисунке 8

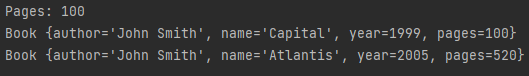


Рисунок 8 - Результат тестирования класса «Book»

## Выводы по работе:

Научился создавать классы, методы и назначать уровень доступа к методам/переменным

# Практическая работа № 20

## Цель работы

Изучить наследование в Java

## 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.

Чтобы объявить один класс наследником от другого, надо использовать после имени класса-наследника ключевое слово extends, после которого идет имя базового класса.

С помощью ключевого слова super мы можем обратиться к любому члену базового класса - методу или полю, если они не определены с модификатором private.

Можно запретить наследование с помощью ключевого слова final.

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.

При определении абстрактных классов используется ключевое слово abstract.

Кроме обычных методов абстрактный класс может содержать абстрактные методы. Такие методы определяются с помощью ключевого слова abstract и не имеют никакого функционала. Производный класс обязан переопределить и реализовать все абстрактные методы, которые имеются в базовом абстрактном классе.

## Выполнение лабораторной работы

### Задание:

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

### Решение:

#### Абстрактный класс «Furniture»

*Листинг 29 - Код "Furniture"*

|  |
| --- |
| abstract class Furniture {  protected String material, manufacturer;  protected int count;  protected double price, width, height, depth, weight;  public Furniture() {  material = "Wood";  manufacturer = "IKEA";  price = 1000;  count = 1;  width = 100;  height = 50;  depth = 100;  weight = 20;  }  public double getWidth() {  return width;  }  public void setWidth(double width) {  this.width = width;  }  public double getHeight() {  return height;  }  public void setHeight(double height) {  this.height = height;  }  public double getWeight() {  return weight;  }  public void setWeight(double weight) {  this.weight = weight;  }  public double getDepth() {  return depth;  }  public void setDepth(double depth) {  this.depth = depth;  }  public String getManufacturer() {  return manufacturer;  }  public void setManufacturer(String manufacturer) {  this.manufacturer = manufacturer;  }  public String getMaterial() {  return material;  }  public void setMaterial(String material) {  this.material = material;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public int getCount() {  return count;  }  public void setCount(int count) {  this.count = count;  }  public abstract String toString();  } |

#### Класс «Table»

*Листинг 30 - Код "Table"*

|  |
| --- |
| public class Table extends Furniture {  private int legs;  private boolean isAngled;  public Table() {  legs = 4;  isAngled = true;  }  public Table(int l, boolean isA) {  legs = l;  isAngled = isA;  }  public int getLegs() {  return legs;  }  public boolean isAngled() {  return isAngled;  }  public void setAngled(boolean angled) {  isAngled = angled;  }  public void setLegs(int legs) {  this.legs = legs;  }  @Override  public String toString() {  return "Table {" +  "legs=" + legs +  ", width=" + width +  ", height=" + height +  ", depth=" + depth +  ", weight=" + weight +  ", angled=" + isAngled +  ", material='" + material + '\'' +  ", manufacturer='" + manufacturer + '\'' +  ", count=" + count +  ", price=" + price +  '}';  }  } |

#### Класс «Chair»

*Листинг 31 - Код "Chair"*

|  |
| --- |
| public class Chair extends Furniture {  private int legs, maxWeight;  private String seatMaterial;  private boolean isRolling;  public Chair() {  legs = 5;  maxWeight = 150;  seatMaterial = "Leather";  isRolling = true;  }  public Chair(int l, int maxW, String sm, boolean isR) {  legs = l;  maxWeight = maxW;  seatMaterial = sm;  isRolling = isR;  }  public int getLegs() {  return legs;  }  public void setLegs(int legs) {  this.legs = legs;  }  public int getMaxWeight() {  return maxWeight;  }  public void setMaxWeight(int maxWeight) {  this.maxWeight = maxWeight;  }  public String getSeatMaterial() {  return seatMaterial;  }  public void setSeatMaterial(String seatMaterial) {  this.seatMaterial = seatMaterial;  }  public boolean isRolling() {  return isRolling;  }  public void setRolling(boolean rolling) {  isRolling = rolling;  }  @Override  public String toString() {  return "Chair {" +  "legs=" + legs +  ", isRolling=" + isRolling +  ", maxWeight=" + maxWeight +  ", main material='" + material + '\'' +  ", seat material='" + seatMaterial + '\'' +  ", manufacturer='" + manufacturer + '\'' +  ", count=" + count +  ", price=" + price +  ", width=" + width +  ", height=" + height +  ", depth=" + depth +  ", weight=" + weight +  '}';  }  } |

#### Класс «Sofa»

*Листинг 32 - Код "Sofa"*

|  |
| --- |
| public class Sofa extends Furniture {  private int doors, sections;  private boolean hangBar;  public Sofa() {  doors = 2;  sections = 4;  hangBar = true;  }  public Sofa(int d, int s, boolean hB) {  doors = d;  sections = s;  hangBar = hB;  }  public int getDoors() {  return doors;  }  public void setDoors(int doors) {  this.doors = doors;  }  public int getSections() {  return sections;  }  public void setSections(int sections) {  this.sections = sections;  }  public boolean isHangBar() {  return hangBar;  }  public void setHangBar(boolean hangBar) {  this.hangBar = hangBar;  }  @Override  public String toString() {  return "Sofa {" +  "doors=" + doors +  ", sections=" + sections +  ", hangBar=" + hangBar +  ", material='" + material + '\'' +  ", manufacturer='" + manufacturer + '\'' +  ", count=" + count +  ", price=" + price +  ", width=" + width +  ", height=" + height +  ", depth=" + depth +  ", weight=" + weight +  '}';  }  } |

#### Класс «FurnitureShop»

*Листинг 33 - Код "FurnitureShop"*

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  import java.util.Scanner;  public class FurnitureShop {  public static class Defaults {  public double width = 100, height = 50, depth = 100, price = 1000;  public String manu = "IKEA", material = "Wood";  public int count = 1;  @Override  public String toString() {  return "Defaults {" +  "width=" + width +  ", height=" + height +  ", depth=" + depth +  ", material='" + material + '\'' +  ", manu='" + manu + '\'' +  ", count=" + count +  ", price=" + price +  '}';  }  }  public static void main(String[] args) {  Defaults dp = new Defaults(); //Для редактирования значений, которые в классе абстракции  List<Furniture> fn = new ArrayList<Furniture>();  Scanner sc = new Scanner(System.in);  String a; //а - выбор, n - для проверки на размер  while (true) {  System.out.println("1) Add new");  System.out.println("2) Show");  System.out.println("3) Change defaults");  System.out.println("4) Exit");  System.out.print("Choose action: ");  a = sc.next();  switch (a) {  case "1" -> {  while (a != "4")  a = AddFurniture(fn, sc, dp); //Костыль, чтобы создать бесконечный цикл для добавления  }  case "2" -> {  ShowFurniture(fn); //Вывод мебели  }  case "3" -> {  ChangeDefaults(dp, sc); //Изменить значения абстракции  }  case "4" -> {  System.exit(0); //Выход  }  }  }  }  private static void ChangeDefaults(Defaults dp, Scanner sc) {  System.out.println(dp);  System.out.print("Width: ");  dp.width = sc.nextDouble();  System.out.print("Height: ");  dp.height = sc.nextDouble();  System.out.print("Depth: ");  dp.depth = sc.nextDouble();  System.out.print("Material: ");  dp.material = sc.next();  System.out.print("Manu: ");  dp.manu = sc.next();  System.out.print("Count: ");  dp.count = sc.nextInt();  System.out.print("Price: ");  dp.price = sc.nextDouble();  }  private static void AddTable(List<Furniture> fn, Scanner sc) {  System.out.println("Legs(number), is angled(1-T, 0-F)");  fn.add(new Table(sc.nextInt(), sc.nextInt() == 1));  }  private static void AddChair(List<Furniture> fn, Scanner sc) {  System.out.println("Legs(n>2), max weight(n>50), seat material, is rolling(1-T, 0-F)");  fn.add(new Chair(sc.nextInt(), sc.nextInt(), sc.next(), sc.nextInt() == 1));  }  private static void AddSofa(List<Furniture> fn, Scanner sc) {  System.out.println("Doors(number), sections(number), has hang bar(1-T, 0-F)");  fn.add(new Sofa(sc.nextInt(), sc.nextInt(), sc.nextInt() == 1));  }  private static void ShowFurniture(List<Furniture> fn) {  for (Furniture furniture : fn) {  System.out.println(furniture);  }  }  static String AddFurniture(List<Furniture> fn, Scanner sc, Defaults dp) {  System.out.println("1) New table");  System.out.println("2) New chair");  System.out.println("3) New sofa");  System.out.println("4) Back");  System.out.print("Choose action: ");  switch (sc.next()) {  case "1" -> {  AddTable(fn, sc);  ApplyDefaults(fn.get(fn.size() - 1), dp);  }  case "2" -> {  AddChair(fn, sc);  ApplyDefaults(fn.get(fn.size() - 1), dp);  }  case "3" -> {  AddSofa(fn, sc);  ApplyDefaults(fn.get(fn.size() - 1), dp);  }  case "4" -> {  return "4";  }  }  return "1";  }  private static void ApplyDefaults(Furniture fn, Defaults dp) {  fn.setWidth(dp.width);  fn.setHeight(dp.height);  fn.setDepth(dp.depth);  fn.setMaterial(dp.material);  fn.setManufacturer(dp.manu);  fn.setCount(dp.count);  fn.setPrice(dp.price);  }  } |

#### Результат:

Главное меню представлено на рисунке 9

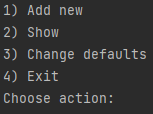


Рисунок 9 - Главное меню

Выбор добавляемого объекта представлен на рисунке 10

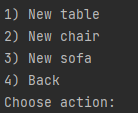


Рисунок 10 - Выбор добавления

Добавление стола представлено на рисунке 11

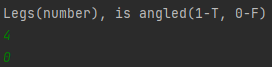


Рисунок 11 - Добавляем стол

Добавление стула представлено на рисунке 12

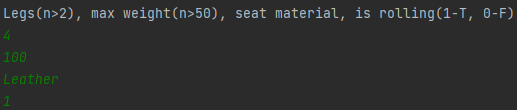


Рисунок 12 - Добавляем стул

Просмотр всех объектов представлен на рисунке 13

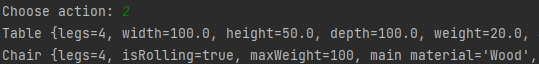


Рисунок 13 - Просмотр объектов

Изменение стандартных (абстрактных) значений представлено на рисунке 14

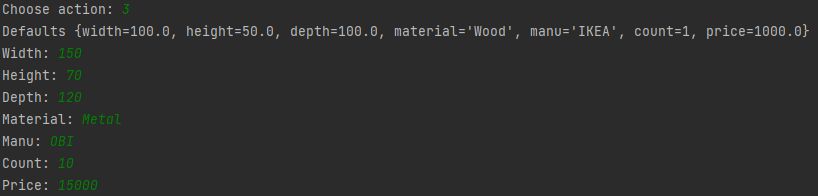


Рисунок 14 - Изменение стандартных значений

## Выводы по работе:

Изучил работу наследования и абстракции в Java.

# Используемая литература

*Список литературы тоже должен быть оформлен по ГОСТ. Первым пунктом можно указать:*

1. *Конспект лекций по дисциплине «Программирование на языке Джава», РТУ МИРЭА, лектор – старший преподаватель Зорина Н.В.*
2. *Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. –СПб.: Питер, 2013. – 240 с.*
3. *Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств – Питер: 2016г. Режим доступа свободный: https://www.htbook.ru/kompjutery\_i\_seti/setevye\_tekhnologii/html5-i-css3-razrabotka-sajtov-dlja-ljubykh-brauzerov-i-ustrojstv;*
4. *Справочник по языку PHP [Электронный ресурс]:php.su— Режим доступа свободный: http://www.php.su;*
5. *И т.д.*