МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения

ОТЧЁТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3
по дисциплине
«Программирование на языке Java»

Тема: Наследование. Абстрактные суперклассы и их подклассы в Java

Выполнил студент группы ИКБО-16-20 Пак С.А.

Принял ассистент кафедры ИиППО Русляков А.А.

Практические работы выполнены «_____» 2021г.

«Зачтено» «_____» 2021г

Москва 2021

СОДЕРЖАНИЕ

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

РЕШЕНИЕ ЗАДАЧИ

1. Постановка задачи

Задания:

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

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

В этом задании, класс Shape определяется как абстрактный класс, который содержит:

- Два поля или переменные класса, объявлены с модификатором protected color (тип String) и filled (тип boolean). Такие защищенные 27 переменные могут быть доступны в подклассах и классах в одном пакете. Они обозначаются со знаком “#” на диаграмме классов в нотации языка UML.
- Методы геттеры и сеттеры для всех переменных экземпляра класса, и метод toString().
- Два абстрактных метода getArea() и getPerimeter() выделены курсивом в диаграмме класса).
- В подклассах Circle (круг) и Rectangle (прямоугольник) должны переопределяться абстрактные методы getArea() и getPerimeter(), чтобы обеспечить их надлежащее выполнение для конкретных экземпляров типа подкласс. Также необходимо для каждого подкласса переопределить toString().

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

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

5. 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).

2. Программный код

Файл Shape.java:

```
package ru.mirea.classes;

public abstract class Shape {
```

```
  protected String color;
  protected boolean filled;

  /**
   * Конструктор по умолчанию
   */
  public Shape() {
    this.color = "";
    this.filled = false;
  }

  /**
   * Конструктор, инициализирующий все поля
   * @param color         цвет фигуры
    * @param filled          параметр, определяющий, закрашена ли
фигура или нет
   */
  public Shape(String color, boolean filled) {
    this.color = color;
    this.filled = filled;
  }

  /**
   * Геттер для поля color
   * @return        цвет фигуры
   */
  public String getColor() {
    return this.color;
  }

  /**
   * Сеттер для поля color
   * @param color         новый цвет фигуры
   */
  public void setColor(String color) {
    this.color = color;
  }

  /**
   * Геттер для поля filled
   * @return        показывает, закращена фигура или нет
   */
  public boolean isFilled() {
    return this.filled;
  }

  /**
```

```
     * Сеттер для поля filled
     * @param filled      новое состояние закращенности фигуры
     */
    public void setFilled(boolean filled) {
      this.filled = filled;
    }

    /**
     * Вычисляет площадь фигуры
     * @return      площадь фигуры
     */
    public abstract double getArea();

    /**
     * Вычисляет периметр фигуры
     * @return      периметр фигуры
     */
    public abstract double getPerimeter();

    /**
     * Объединяет информацию об объекте в одну строку
     * @return      строка с инфомацией об объекте
     */
    @Override
    public abstract String toString();
}
```

Файл Circle.java:

```
package ru.mirea.classes;

public class Circle extends Shape {
    protected double radius;

    /**
     * Конструктор по умолчанию
     */
    public Circle() {
      this.radius = 0.0;
    }

    /**
     * Конструктор, создающий окружность
     * @param radius      радиус окружности
     */
    public Circle(double radius) {
      this.radius = radius;
```

```
}

/**
 * Конструктор, создающий закрашенный круг
 * @param radius        радиус окружности
 * @param color         цвет линии окружности
 * @param filled        закрашенность
 */
public Circle(double radius, String color, boolean filled) {
  this.radius = radius;
  this.color = color;
  this.filled = filled;
}

/**
 * Геттер для поля radius
 * @return        радиус окружности
 */
public double getRadius() {
  return this.radius;
}

/**
 * Сеттер для поля radius
 * @param radius      новый радиус окружности
 */
public void setRadius(double radius) {
  this.radius = radius;
}

/**
 * Вычисляет площадь круга
 * @return      площадь круга
 */
public double getArea() {
  return Math.PI * this.radius * this.radius;
}

/**
 * Вычисляет длину окружности
 * @return      длина окружности
 */
public double getPerimeter() {
  return 2.0 * Math.PI * this.radius;
}

/**
```

```
   * Объединяет информацию об объекте в одну строку
   * @return      строка с инфомацией об объекте
   */
  @Override
  public String toString() {
    return "Circle {\n"
         + "  radius: " + this.radius + "\n"
         + "  color: " + this.color + "\n"
         + "  filled: " + this.filled + "\n"
         + "  area: " + this.getArea() + "\n"
         + "  perimeter: " + this.getPerimeter() + "\n"
         + "}";
  }
}
```

Файл Rectangle.java:

```
package ru.mirea.classes;

public class Rectangle extends Shape {
  protected double width;
  protected double length;

  /**
   * Конструктор по умолчанию
   */
  public Rectangle() {
    this.width = 0.0;
    this.length = 0.0;
  }

  /**
   * Конструктор, создающий форму прямоугольника
   * @param width         ширина
   * @param length        длина
   */
  public Rectangle(double width, double length) {
    this.width = width;
    this.length = length;
  }

  /**
   * Конструктор, создающий цветной прямоугольник
   * @param width       ширина
   * @param length      длина
   * @param color       цвет
   * @param filled      заливка
```

```
    */
    public Rectangle(double width, double length, String color,
boolean filled) {
     this.width = width;
     this.length = length;
     this.color = color;
     this.filled = filled;
   }

   /**
    * Геттер для поля width
    * @return    ширина прямоугольника
    */
   public double getWidth() {
     return this.width;
   }

   /**
    * Сеттер для поля width
    * @param width     новая ширина прямоугольника
    */
   public void setWidth(double width) {
     this.width = width;
   }

   /**
    * Геттер для поля length
    * @return    длина прямоугольника
    */
   public double getLength() {
     return this.length;
   }

   /**
    * Сеттер для поля length
    * @param length    длина прямоугольника
    */
   public void setLength(double length) {
     this.length = length;
   }

   /**
    * Вычисляет площадь прямоугольника
    * @return      площадь прямоугольника
    */
   public double getArea() {
     return this.width * this.length;
```

```
  }

  /**
   * Вычисляет периметр прямоугольника
   * @return      периметр прямоугольника
   */
  public double getPerimeter() {
    return 2.0 * (this.width + this.length);
  }

  /**
   * Объединяет информацию об объекте в одну строку
   * @return      строка с инфомацией об объекте
   */
  @Override
  public String toString() {
    return "Rectangle {\n"
         + "\twidth: " + this.width + "\n"
         + "\tlength: " + this.length + "\n"
         + "\tcolor: " + this.color + "\n"
         + "\tfilled: " + this.filled + "\n"
         + "\tarea: " + this.getArea() + "\n"
         + "\tperimeter: " + this.getPerimeter() + "\n"
         + "}";
  }
}
```

Файл Square.java:

```
package ru.mirea.classes;

public class Square extends Rectangle {
  /**
   * Конструктор по умолчанию
   */
  public Square() {
    this.width = 0.0;
    this.length = 0.0;
  }

  /**
   * Конструктор, создающий форму квадрата
   * @param side      длина стороны квадрата
   */
  public Square(double side) {
    this.width = side;
    this.length = side;
```

```
}

/**
 * Конструктор, создающий закращенный квадрат
 * @param side        длина стороны
 * @param color       цвет квадрата
 * @param filled      заливка
 */
public Square(double side, String color, boolean filled) {
  this.width = side;
  this.length = side;
  this.color = color;
  this.filled = filled;
}

/**
 * Геттер для поля width или length
 * @return      длина стороны квадрата
 */
public double getSide() {
  return this.width;
}

/**
 * Сеттер для поля side
 * @param side      новая длина стороны квадрата
 */
public void setSide(double side) {
  this.width = side;
  this.length = side;
}

/**
 * Сеттер для поля width
 * @param width     новая ширина прямоугольника
 */
@Override
public void setWidth(double width) {
  this.setSide(width);
}

/**
 * Сеттер для поля length
 * @param length    длина прямоугольника
 */
@Override
public void setLength(double length) {
```

```
    this.setSide(length);
  }

  @Override
  public String toString() {
    return "Square {\n"
         + "\twidth: " + this.width + "\n"
         + "\tlength: " + this.length + "\n"
         + "\tcolor: " + this.color + "\n"
         + "\tfilled: " + this.filled + "\n"
         + "\tarea: " + this.getArea() + "\n"
         + "\tperimeter: " + this.getPerimeter() + "\n"
         + "}";
  }
}
```

Файл App.java:

```
package ru.mirea;

import java.util.Scanner;
import ru.mirea.classes.*;

public class App {
  private static final Scanner IN = new Scanner(System.in);

  public static void main(String[] args) {
    /*======== КРУГ ========*/
    System.out.print("Введите радиус окружности: ");
    double circleRadius = IN.nextDouble();

    System.out.print("Введите цвет круга: ");
    String circleColor = IN.next();

    Shape circle = new Circle(circleRadius, circleColor, false);
    /*======== КРУГ ========*/

    System.out.println();

    /*======== ПРЯМОУГОЛЬНИК ========*/
    System.out.print("Введите ширину прямоугольника: ");
    double recWidth = IN.nextDouble();

    System.out.print("Введите длину прямоугольника: ");
    double recLength = IN.nextDouble();

    System.out.print("Введите цвет прямоугольника: ");
    String recColor = IN.next();
```

```
     Rectangle rec = new Rectangle(recWidth, recLength, recColor,
true);
    /*======== ПРЯМОУГОЛЬНИК ========*/

    System.out.println();

    /*======== КВАДРАТ ========*/
    System.out.print("Введите длину стороны квадрата: ");
    double squareSide = IN.nextDouble();

    System.out.print("Введите цвет квадрата: ");
    String squareColor = IN.next();

    Square square = new Square(squareSide, squareColor, true);
    /*======== КВАДРАТ ========*/

    System.out.println();
    System.out.println(circle);
    System.out.println(rec);
    System.out.println(square);
  }
}
```

Файл Movable.java:

```
package ru.mirea.interfaces;

public interface Movable {
  /**
   * Перемещает точку вверх по координатной плоскости
   */
  void moveUp();

  /**
   * Перемещает точку вниз по координатной плоскости
   */
  void moveDown();

  /**
   * Перемещает точку влево по координатной плоскости
   */
  void moveLeft();

  /**
   * Перемещает точку вправо по координатной плоскости
   */
  void moveRight();
}
```

Файл MovablePoint.java:

```
package ru.mirea.classes;

import ru.mirea.interfaces.*;

public class MovablePoint implements Movable {
  protected int x;
  protected int y;
  protected int xSpeed;
  protected int ySpeed;

  /**
   * Конструктор, создающий движущуюсю точку
   * @param x              координата по оси абсцисс
   * @param y              координата по оси ординат
   * @param xSpeed         скорость по оси абсцисс
   * @param ySpeed         скорость по оси ординат
   */
  public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
    this.x = x;
    this.y = y;
    this.xSpeed = xSpeed;
    this.ySpeed = ySpeed;
  }

  /**
   * Объединяет информацию об объекте в одну строку
   * @return      строка с инфомацией об объекте
   */
  @Override
  public String toString() {
    return "MovablePoint {\n"
         + "\tx: " + this.x + "\n"
         + "\ty: " + this.y + "\n"
         + "\txSpeed: " + this.xSpeed + "\n"
         + "\tySpeed: " + this.ySpeed + "\n"
         + "}";
  }

  /**
   * Перемещает точку вверх по координатной плоскости
   */
  public void moveUp() {
    this.y += this.ySpeed;
  }
```

```
  /**
   * Перемещает точку вниз по координатной плоскости
   */
  public void moveDown() {
    this.y -= this.ySpeed;
  }

  /**
   * Перемещает точку влево по координатной плоскости
   */
  public void moveLeft() {
    this.x -= this.xSpeed;
  }

  /**
   * Перемещает точку вправо по координатной плоскости
   */
  public void moveRight() {
    this.x += this.xSpeed;
  }
}
```

Файл MovableCircle.java:

```
package ru.mirea.classes;

import ru.mirea.interfaces.*;

public class MovableCircle implements Movable {
  private int radius;
  private MovablePoint center;

  /**
   * Создаёт движущийся круг
   * @param x           координата центра по оси абсцисс
   * @param y           координата центра по оси ординат
   * @param xSpeed      скорость по оси абсцисс
   * @param ySpeed      скорость по оси ординат
   * @param radius      радиус окружности
   */
   public MovableCircle(int x, int y, int xSpeed, int ySpeed, int
radius) {
    this.radius = radius;
    this.center = new MovablePoint(x, y, xSpeed, ySpeed);
  }

  /**
   * Объединяет информацию об объекте в одну строку
```

```
     * @return      строка с инфомацией об объекте
     */
    @Override
    public String toString() {
      return "MovableCircle {\n"
           + "\tx: " + this.center.x + "\n"
           + "\ty: " + this.center.y + "\n"
           + "\txSpeed: " + this.center.xSpeed + "\n"
           + "\tySpeed: " + this.center.ySpeed + "\n"
           + "\tradius: " + this.radius + "\n"
           + "}";
    }

    /**
     * Перемещает точку вверх по координатной плоскости
     */
    public void moveUp() {
      this.center.moveUp();
    }

    /**
     * Перемещает точку вниз по координатной плоскости
     */
    public void moveDown() {
      this.center.moveDown();
    }

    /**
     * Перемещает точку влево по координатной плоскости
     */
    public void moveLeft() {
      this.center.moveLeft();
    }

    /**
     * Перемещает точку вправо по координатной плоскости
     */
    public void moveRight() {
      this.center.moveRight();
    }
}
```

Файл MovableRectangle.java:

```
package ru.mirea.classes;

import ru.mirea.interfaces.*;
```

```
public class MovableRectangle implements Movable {
  private MovablePoint topLeft;
  private MovablePoint bottomRight;

  /**
   * Создаёт движущийся прямоугольник
   * @param x1          координата левой верхней точки по оси X
   * @param y1          координата левой верхней точки по оси Y
   * @param x2          координата правой нижней точки по оси X
   * @param y2          координата правой нижней точки по оси Y
   * @param xSpeed      скорость по оси X
   * @param ySpeed      скорость по оси Y
   */
  public MovableRectangle(
    int x1,
    int y1,
    int x2,
    int y2,
    int xSpeed,
    int ySpeed
  ) {
    this.topLeft = new MovablePoint(x1, y1, xSpeed, ySpeed);
    this.bottomRight = new MovablePoint(x2, y2, xSpeed, ySpeed);
  }

  /**
   * Объединяет информацию об объекте в одну строку
   * @return      строка с инфомацией об объекте
   */
  @Override
  public String toString() {
    return "MovableRectangle {\n"
         + "\tx1: " + this.topLeft.x + "\n"
         + "\ty1: " + this.topLeft.y + "\n"
         + "\tx2: " + this.bottomRight.x + "\n"
         + "\ty2: " + this.bottomRight.y + "\n"
         + "\txSpeed: " + this.topLeft.xSpeed + "\n"
         + "\tySpeed: " + this.bottomRight.ySpeed + "\n"
         + "}";
  }

  /**
   * Перемещает точку вверх по координатной плоскости
   */
  public void moveUp() {
    this.topLeft.moveUp();
    this.bottomRight.moveUp();
```

```
  }

  /**
   * Перемещает точку вниз по координатной плоскости
   */
  public void moveDown() {
    this.topLeft.moveDown();
    this.bottomRight.moveDown();
  }

  /**
   * Перемещает точку влево по координатной плоскости
   */
  public void moveLeft() {
    this.topLeft.moveLeft();
    this.bottomRight.moveLeft();
  }

  /**
   * Перемещает точку вправо по координатной плоскости
   */
  public void moveRight() {
    this.topLeft.moveRight();
    this.bottomRight.moveRight();
  }

  /**
   * Проверяет, одинаковы ли скорости точек
   */
  public boolean checkTheSameSpeed() {
    return (this.topLeft.xSpeed == this.bottomRight.xSpeed)
        && (this.topLeft.ySpeed == this.bottomRight.ySpeed);
  }
}
```

3. Вывод программы

В программе пользователь поочерёдно вводит значения для полей объектов, и в конце выводится вся информация о них (рис.1).

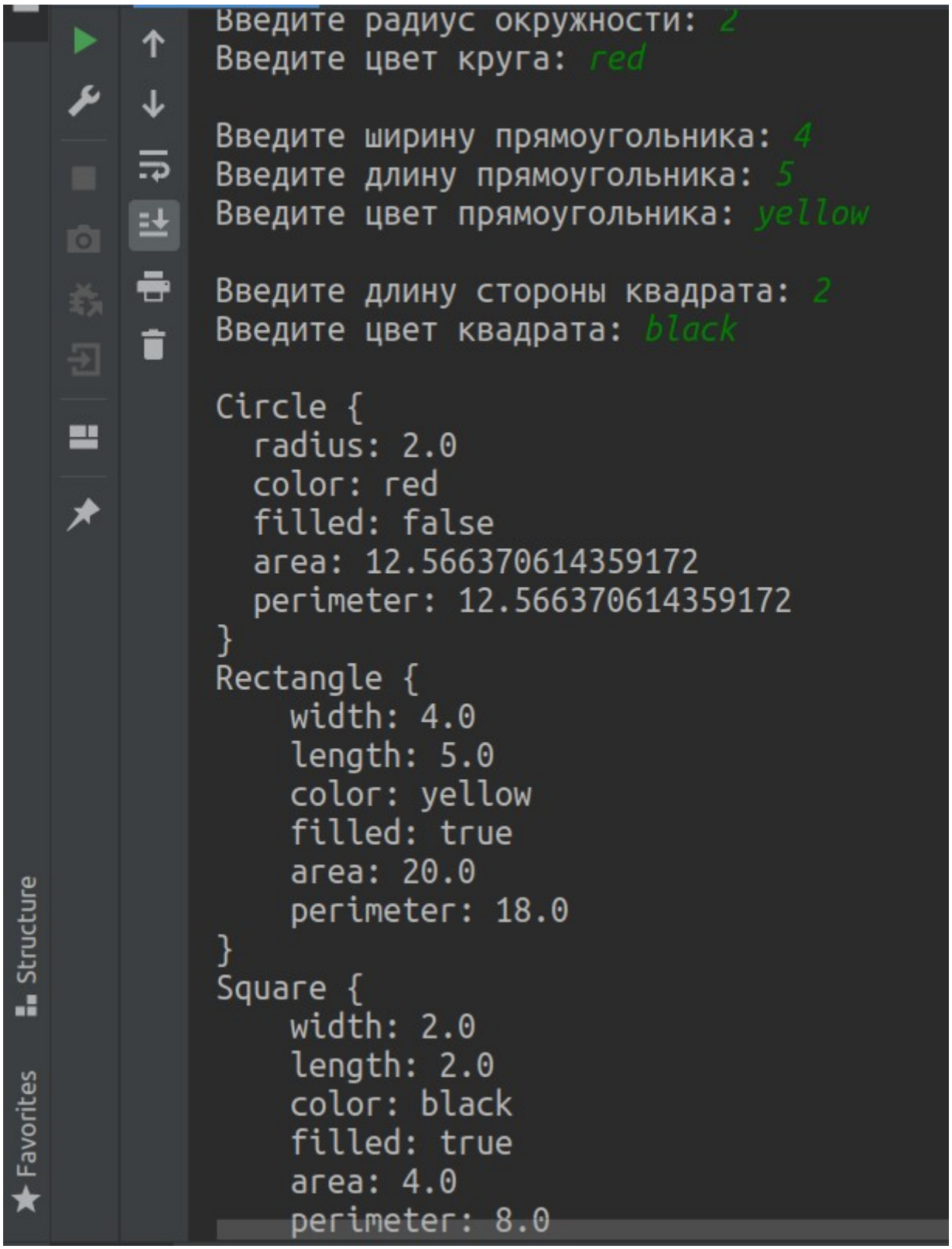

Рис.1 Вывод программы

ВЫВОД

В ходе выполнения работы освоил на практике работу с абстрактными классами и наследованием на Java.