**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное образовательное бюджетное учреждение высшего образования**

**«Вологодский государственный университет»**

**(ВоГУ)**

* 1. **ОБЪЕКТНО-ОРЕНТИРОВАННОЕ программирование**
  2. **лабораторнАЯ РАБОТА №4**

**«Создание классов на языке Java»**

**Направление подготовки: 09.03.01 Информатика и вычислительная техника**

**Направленность (профиль): Программное обеспечение средств вычислительной техники и автоматизированных систем**

**Форма обучения: очная**

**Институт: Математики, естественных и компьютерных наук**

**Кафедра: Автоматики и вычислительной техники**

**Группа: 4Б09 ПО-41**

**Студент: Махонин М.Н.**

**Руководитель: Сорокин А.Н.**

Вологда

2021 г.

**Цель работы:** научиться разрабатывать классы на языке Java.

**Вариант 2**

1. Создать базовый класс «фигура» (абстрактный класс или интерфейс) с виртуальными функциями: площадь поверхности, объем, ввод и вывод информации о фигуре.

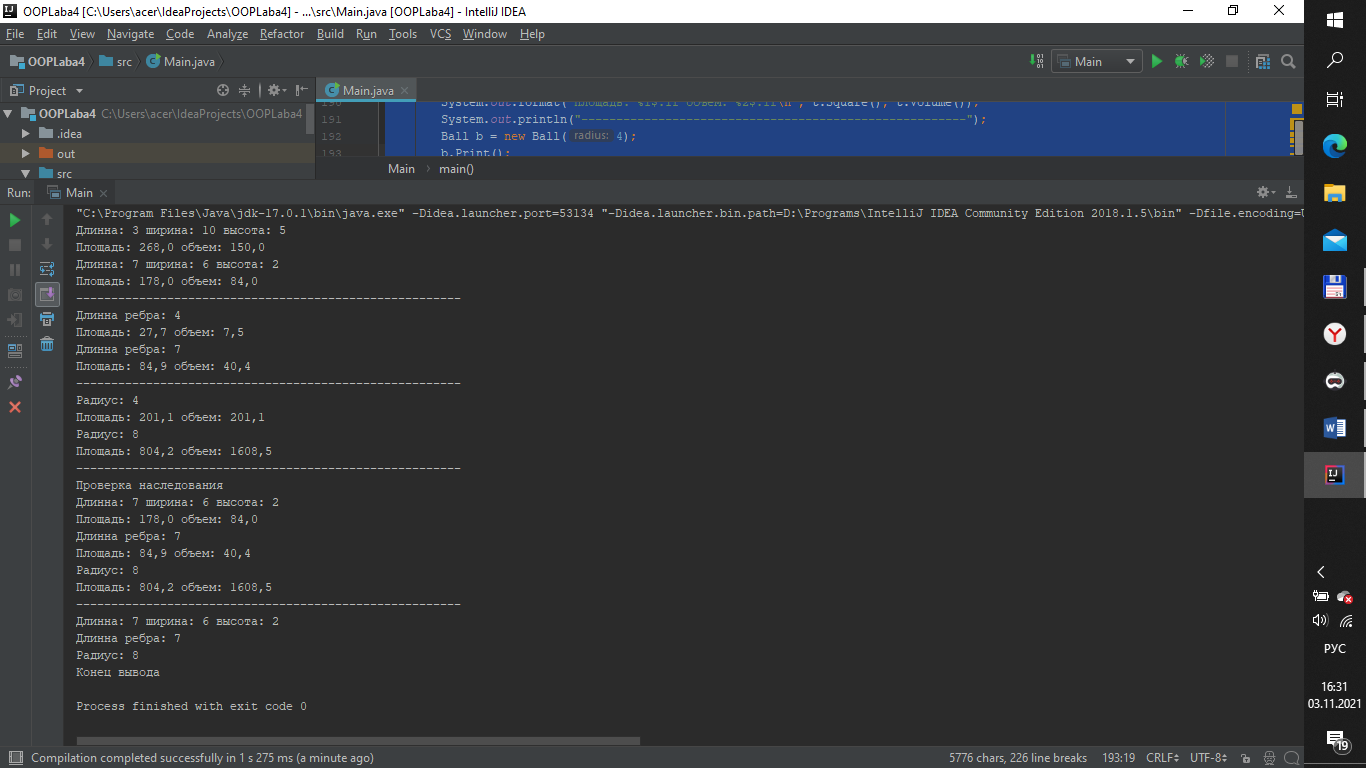
* Создать производные классы: параллелепипед, тетраэдр, шар.
* В производных классах определить: скрытые поля, конструктор по умолчанию, конструктор с параметрами, методы для установки и получения значение полей, методы для расчета площади и объема, ввода и вывода информации.

2. Разработать программу, которая демонстрировала бы работоспособность всех методов классов. Для проверки определить коллекцию ссылок на базовый класс, которым присваиваются адреса различных объектов.

3. Разработать программу, которая демонстрировала бы возможности сериализации. Для проверки введите данные с клавиатуры, сохраните информацию в файл (требуется сохранить несколько объектов разных классов из задания 1), затем прочитайте данные в цикле и выведите информацию на экран (в этом цикле неизвестно количество и порядок объектов в файле).

import java.io.\*;  
  
  
interface Shape {  
 public double Square();  
 public double Volume();  
 public void InputData(int ... data);  
 public void Print();  
}  
  
class Parallelepiped implements Shape, Serializable {  
 int length;  
 int height;  
 int width;  
  
 public Parallelepiped(){  
 length = 1;  
 height = 1;  
 width = 1;  
 }  
  
 public Parallelepiped(int length, int heigth, int width){  
 this.length = length;  
 this.height = heigth;  
 this.width = width;  
 }  
  
 public void SetLength(int value){  
 if(value <= 0 ){  
 System.*out*.println("Неверное значение");  
 return;  
 }  
 length = value;  
 }  
  
 public void SetHeight(int value){  
 if(value <= 0 ){  
 System.*out*.println("Неверное значение");  
 return;  
 }  
 height = value;  
 }  
  
 public void SetWidth(int value){  
 if(value <= 0 ){  
 System.*out*.println("Неверное значение");  
 return;  
 }  
 width = value;  
 }  
  
 public int GetLength(){  
 return length;  
 }  
  
 public int GetHeight(){  
 return height;  
 }  
  
 public int GetWidth(){  
 return width;  
 }  
  
 public double Square(){  
 return 2 \* Math.*pow*(length, 2) + 2 \* Math.*pow*(height, 2) + 2 \* Math.*pow*(width, 2);  
 }  
  
 public double Volume(){  
 return length \* width \* height;  
 }  
  
 public void Print(){  
 System.*out*.format("Длинна: %d ширина: %d высота: %d\n", length, width, height);  
 }  
  
 public void InputData(int ... data){  
 if(data.length < 3){  
 System.*out*.format("Неверное количество аргументов\n");  
 return;  
 }  
 this.length = data[0];  
 this.height = data[1];  
 this.width = data[2];  
 }  
}  
  
class Tetraider implements Shape, Serializable{  
 int edge;  
  
 public Tetraider(){  
 edge = 1;  
 }  
  
 public Tetraider(int edge){  
 this.edge = edge;  
 }  
  
 public void SetEdge(int value){  
 if(value <= 0 ){  
 System.*out*.println("Неверное значение");  
 return;  
 }  
 edge = value;  
 }  
  
 public int GetEdge(){  
 return edge;  
 }  
  
 public double Square(){  
 return Math.*sqrt*(3) \* Math.*pow*(edge, 2);  
 }  
  
 public double Volume(){  
 return (Math.*sqrt*(2) / 12.0) \* Math.*pow*(edge, 3);  
 }  
  
 public void Print(){  
 System.*out*.format("Длинна ребра: %d\n", edge);  
 }  
  
 public void InputData(int ... data){  
 if(data.length < 1){  
 System.*out*.format("Неверное количество аргументов\n");  
 return;  
 }  
 edge = data[0];  
 }  
}  
  
class Ball implements Shape, Serializable  
{  
 int radius;  
  
 public Ball(){  
 radius = 1;  
 }  
  
 public Ball(int radius){  
 this.radius = radius;  
 }  
  
 public void SetRadius(int value){  
 if(value <= 0){  
 System.*out*.println("Неверное значение");  
 return;  
 }  
 radius = value;  
 }  
  
 public int GetRadius(){  
 return radius;  
 }  
  
 public double Square(){  
 return 4 \* Math.*PI* \* Math.*pow*(radius, 2);  
 }  
  
 public double Volume(){  
 return (4 / 3) \* Math.*PI* \* Math.*pow*(radius, 3);  
 }  
  
 public void Print(){  
 System.*out*.format("Радиус: %d\n", radius);  
 }  
  
 public void InputData(int ... data){  
 if(data.length < 1){  
 System.*out*.format("Неверное количество аргументов\n");  
 return;  
 }  
 radius = data[0];  
 }  
}  
public class Main  
{  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
 Parallelepiped p = new Parallelepiped(3, 5, 10);  
 p.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", p.Square(), p.Volume());  
 p.InputData(7, 2, 6);  
 p.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", p.Square(), p.Volume());  
 System.*out*.println("-------------------------------------------------------");  
 Tetraider t = new Tetraider(4);  
 t.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", t.Square(), t.Volume());  
 t.InputData(7);  
 t.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", t.Square(), t.Volume());  
 System.*out*.println("-------------------------------------------------------");  
 Ball b = new Ball(4);  
 b.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", b.Square(), b.Volume());  
 b.InputData(8);  
 b.Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", b.Square(), b.Volume());  
 System.*out*.println("-------------------------------------------------------");  
 System.*out*.println("Проверка наследования");  
 Shape[] array = new Shape[3];  
 array[0] = p;  
 array[1] = t;  
 array[2] = b;  
 for(int i = 0; i < 3; i++){  
 array[i].Print();  
 System.*out*.format("Площадь: %1$.1f объем: %2$.1f\n", array[i].Square(), array[i].Volume());  
 }  
 System.*out*.println("-------------------------------------------------------");  
 FileOutputStream fos = new FileOutputStream("test.out");  
 ObjectOutputStream oos = new ObjectOutputStream(fos);  
 for(int i = 0; i < 3; i++){  
 oos.writeObject((array[i]));  
 }  
 oos.flush();  
 oos.close();  
 FileInputStream fis = new FileInputStream("test.out");  
 ObjectInputStream oin = new ObjectInputStream(fis);  
 try{  
 while (true){  
 Shape s = (Shape)oin.readObject();  
 s.Print();  
 }  
 }catch (Exception e){  
 System.*out*.println("Конец вывода");  
 }  
 }  
}

**Вывод:**



**Контрольные вопросы**

1. **Что понимается под термином «абстрактный класс»? В каких случаях применяются абстрактные классы?**

Абстрактный класс в объектно-ориентированном программировании — базовый класс, который не предполагает создания экземпляров. Абстрактные классы реализуют на практике один из принципов ООП — полиморфизм. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объём и наименьшее содержание.

1. **Что понимается под термином «интерфейс»? В каких случаях применяются интерфейсы?**

Интерфе́йс — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое поведенческое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации.

Использование интерфейсов — один из вариантов обеспечения полиморфизма в объектных языках и средах. Все классы, реализующие один и тот же интерфейс, с точки зрения определяемого ими поведения, ведут себя внешне одинаково. Это позволяет писать обобщённые алгоритмы обработки данных, использующие в качестве типов параметры интерфейсов, и применять их к объектам различных типов, всякий раз получая требуемый результат.