**《Java程序设计》上机作业2**

|  |  |
| --- | --- |
| **学号：** | **202103151422** |
| **姓名：** | **温家伟** |
| **班级：** | **大数据分析班** |
| **日期：** | **2023.10.16** |

**一、目的**

通过作业掌握下列知识:

* 熟悉类和对象的概念，掌握类与对象的关系；
* 掌握对象的引用及内存存储情况；
* 掌握类的设计方法和对象成员的引用方法等。

**二、内容**

**1. 请根据以下类的设计要求编写Java程序。**

- 类名：**Birth**

- 成员变量：year、month、day

- 构造方法：

无参构造方法将成员变量初始化成: year=2000, month=1, day=1(对应日期为2000年1月1日)；

带参构造方法对成员变量进行合理初始化，需要对年月日对应值的范围作简单的控制。

- 成员方法：String toString()，返回对象的出生年月日信息。

要求创建Brith对象，通过构造方法初始化成员变量，输出对象的出生年月日信息及年龄。引入java.util. Calendar类，创建Calendar对象(用Calendar.getInstance()产生Calendar对象)，调用get(Calendar.YEAR)获取当前的年份值。

|  |
| --- |
| * **问题分析**   根据题目要求，需要创建一个类名为Birth的类，该类包含成员变量year、month、day，以及构造方法和成员方法。构造方法有无参构造方法和带参构造方法，成员方法是toString()方法，用于返回出生年月日信息。此外，还需要引入Calendar类来获取当前年份值，计算年龄。 |
| * **代码实现**   import java.util.Calendar; public class Birth {  int \_year;  int \_month;  int \_day;  // 无参构造  public Birth(){  \_year = 2000;  \_month = 1;  \_day = 1;  }  public Birth(int year, int month, int day) {  // 简单控制日期范围  boolean flag = false;  if (year <= 0) flag = true;  if (month <= 0 || month > 12) flag = true;  if ((month == 1 || month == 3 || month ==5 || month ==7 || month == 8 || month == 10 || month == 12) &&  (day <= 0 || day > 31)) flag = true;  if ((month == 2) && (day >= 29 || day <=0)) flag = true;  if ((month == 4 || month == 6 || month == 9 || month == 11) && (day <= 0 || day > 30)) flag = true;  if (flag) {  System.*out*.println("日期有误!!!");  }  \_year = year;  \_month = month;  \_day = day;  }  public String toString() {  String ret = "";  ret += Integer.*toString*(\_year);  ret += "年";  ret += Integer.*toString*(\_month);  ret += "月";  ret += Integer.*toString*(\_day);  ret += "日";  return ret;  }   public int calage() {  Calendar today = Calendar.*getInstance*();  int gap\_year = today.get(Calendar.*YEAR*) - \_year;  int month = today.get(Calendar.*MONTH*);  int day = today.get(Calendar.*DATE*);  //  if (month > \_month) {  return gap\_year;  }  else if (month == \_month) {  if (day >= \_day) return gap\_year;  else return gap\_year - 1;  }  else {  return gap\_year - 1;  }  }   public static void main(String[] args) {  Birth birth = new Birth(2002, 11, 13);  System.*out*.print("出生日期：");  System.*out*.println(birth.toString());  System.*out*.print("现在");  System.*out*.println(birth.calage() + "岁");  } } |
| * **运行结果** |
| * **思考及总结**   **要计算年龄，可以通过获取当前年份值和出生年份进行计算，并注意考虑到当前日期是否已过生日的情况。** |

**2.** 编写一个汽车类Car。

Car具有以下属性：

品牌：brand ，类型为String；

发动机排量：engineDisplacement，类型为double；

速度：speed，类型为double；

状态：status，类型为boolean；

最高时速：maxSpeed，类型为double。

Car具有以下方法：

构造方法：Car(String brand, double engineDisplacement, double maxSpeed)，该方法使用参数设置成员变量的值。

启动：start( )，该方法使得status的值变成true;

加速：speedUp( )，当汽车处于启动状态时，该方法每调用一次，速度增加5，但速度不得高于最高时速。

减速：slowDown( )，当汽车处于启动状态时，该方法每调用一次，速度减5，但速度不得小于0。

熄火：stop( )，当speed为零时，将status的值变成false。

每个方法除了改变成员变量的值之外，还要打印出方法执行后的状态和速度。

编写main方法实例化一个Car对象，调用以上方法，输出相关信息。

|  |
| --- |
| * **问题分析** * 题目要求构建一个汽车类，包括属性和方法，需要实现汽车启动、加速、减速和熄火等功能，操作需要考虑汽车当前状态和速度。 |
| * **代码实现**   public class Car {  String \_brand;  double \_engineDisplacement;  double \_speed;  boolean \_status;  double \_maxSpeed;   public Car(String brand, double engineDisplacement, double maxSpeed) {  \_brand = brand;  \_engineDisplacement = engineDisplacement;  \_maxSpeed = maxSpeed;   \_status = false;  \_speed = 0;   }  public void start() {  \_status = true;  }  public void speedUp() {  if (\_status == true && \_speed <= \_maxSpeed) {  \_speed += 5;  // 速度最多增到\_maxSpeed  \_speed = \_speed > \_maxSpeed ? \_maxSpeed : \_speed;  }  }  public void showDown() {  if (\_status == true && \_speed >= 0) {  \_speed -= 5;  // 速度最多减为零  \_speed = \_speed < 0 ? 0 : \_speed;  }  }  public void stop() {  if (\_speed == 0) {  \_status = false;  }  }  public void Print() {  System.*out*.println("状态：" + \_status);  System.*out*.println("速度：" + \_speed);  }   public static void main(String[] args) {  Car car = new Car("宝马", 2.5, 180.0);  System.*out*.println("汽车启动：");  car.start();  car.Print();   System.*out*.println("汽车加速：");  for (int i = 0; i < 10; i++) {  car.speedUp();  car.Print();  }   System.*out*.println("汽车减速：");  for (int i = 0; i < 10; i++) {  car.showDown();  car.Print();  }   System.*out*.println("熄火：");  car.stop();  car.Print();  } } |
| * **运行结果** |
| * **思考及总结**  1. **对于属性品牌、发动机排量、最高时速可以使用私有修饰符，以保证数据的封装性。** |

**3.** **请设计一个程序。**其中至少包含四个类，分别是三角形类（Triangle）、正方形类（Rectangle）、圆类（Circle）和正五边形类（Regular Pentagon）。要求：

- 每个类都具有构造方法，能够构造任意大小的图形。例如：通过三角形类的构造方法可以构造出任意三边长的三角形（无需判断是否能构成三角形）

- 每个类都含有计算该图形周长的成员方法和计算该图形面积的成员方法

- 在上述四个类之外，编写一个主类。在主方法中创建这些类的实例对象，并输出这些实例对象的周长和面积。在创建这些实例对象时，构造方法的调用参数可以自行设计。

Tips:

1. 任意边长三角形计算面积公式：

1. 正五边形的面积计算公式：

S=1.72×a²

1. 开根号运算的实现方式：

 🡨 Math.sqrt(va)

|  |
| --- |
| * **问题分析**   题目要求设计四个类，分别是三角形类（Triangle）、正方形类（Rectangle）、圆类（Circle）和正五边形类（Regular Pentagon）。每个类都需要具有构造方法、计算周长的成员方法和计算面积的成员方法。另外还需要编写一个主类，创建这些类的实例对象，并输出它们的周长和面积。 |
| * **代码实现**   import java.math.MathContext; public class test {  public static void main(String[] args) {  Triangle triangle = new Triangle(3, 4, 5);  System.*out*.println("面积：" + triangle.area());  System.*out*.println("周长：" + triangle.Perimeter());   Rectangle rectangle = new Rectangle(3);  System.*out*.println("面积：" + rectangle.area());  System.*out*.println("周长：" + rectangle.Perimeter());   Circle circle = new Circle(3);  System.*out*.println("面积：" + circle.area());  System.*out*.println("周长：" + circle.Perimeter());   RegularPentagon regularPentagon = new RegularPentagon(3);  System.*out*.println("面积：" + regularPentagon.area());  System.*out*.println("周长：" + regularPentagon.Perimeter());  } }  class Triangle {  double \_a;  double \_b;  double \_c;   public Triangle(double a, double b, double c) {  \_a = a;  \_b = b;  \_c = c;  }  public double area() {  return Math.*sqrt*((\_a + \_b + \_c) \* (\_a + \_b - \_c) \* (\_a + \_c - \_b) \* (\_b + \_c - \_a)) / 4;  }  public double Perimeter() {  return \_a + \_b + \_c;  } }  class Rectangle {  double \_a;  public Rectangle(double a) {  \_a = a;  }  public double area() {  return \_a \* \_a;  }  public double Perimeter() {  return 4 \* \_a;  } }  class Circle {  double \_r;  public Circle(double r) {  \_r = r;  }  public double area() {  return Math.*PI* \* \_r \* \_r;  }  public double Perimeter() {  return 2 \* Math.*PI* \* \_r;  } }  class RegularPentagon {  double \_a;  public RegularPentagon(double a) {  \_a = a;  }  public double area() {  return 1.72 \* \_a \* \_a;  }  public double Perimeter() {  return 5 \* \_a;  } } |
| * **运行结果** |
| * **思考及总结**   在主类中，可以创建每个图形类的实例对象，并调用其成员方法来计算周长和面积，最后输出结果。 |

**4.** 编写一个复数类Complex验证两个复数1+2\*i和3+4\*i相加产生一个新的复数4+6\*i。复数类Complex必须满足如下要求：

（1）复数类Complex的属性有：

realPart是int型，代表复数的实数部分

imaginPart是int型，代表复数的虚数部分。

（2）复数类Complex的方法有：

Complex()构造方法，将复数的实部和虚数都置0

Complex(int r, int i)构造方法，形参r为实部的初值，i为虚部的初值。

Complex complexAdd(Complex a)将当前复数对象与形参复数对象相加，所得的结果仍是一个复数值，将其返回给此方法的调用者。

Complex complexAdd(Complex a, Complex b) 可选

String toString()把当前复数对象的实部、虚部组合成a+b\*i的字符串形式，其中，a和b分别为实部和虚部的数据。

|  |
| --- |
| * **问题分析**   题目要求编写一个复数类Complex，验证两个复数1+2i和3+4i相加得到4+6\*i。Complex类需要具有实数部分和虚数部分的属性，以及构造方法、复数相加方法和字符串表示方法。 |
| * **代码实现**   public class Complex {  int \_realPart;  int \_imaginPart;   Complex() {  \_realPart = 0;  \_imaginPart = 0;  }  Complex(int r, int i){  \_realPart = r;  \_imaginPart = i;  }  public Complex complexADD(Complex a) {  int r = a.\_realPart;  int i = a.\_imaginPart;  return new Complex(r + \_realPart, i + \_imaginPart);  }  public static Complex complexADD(Complex a, Complex b) {  int r1 = a.\_realPart;  int i1 = a.\_imaginPart;  int r2 = b.\_realPart;  int i2 = b.\_imaginPart;  return new Complex(r1 + r2, i1 + i2);  }  public String toString(Complex com) {  String ret = "";  ret += Integer.*toString*(com.\_realPart);  ret += "+";  ret += Integer.*toString*(com.\_imaginPart);  ret += "\*i";  return ret;  }  public static void main(String[] args) {  Complex com1 = new Complex(1, 2);  Complex com2 = new Complex(3, 4);  Complex ret = Complex.*complexADD*(com1, com2);  System.*out*.println("测试两个参数的complexADD");  System.*out*.println(ret.toString(ret));  ret = com1.complexADD(com2);  System.*out*.println("测试一个参数的complexADD");  System.*out*.println(ret.toString(ret));  } } |
| * **运行结果** |
| * **思考及总结**   构造方法Complex()用于初始化复数对象，将实数部分和虚数部分都置为0。 |

**5.** 编写一个银行账户类Account，并编写一个main方法，对Account类进行测试，在main方法中要调用Account的所有方法。存取款的步骤：先存入帐号若干元（随机产生），然后取款若干元（随机产生），最后查询帐号信息。

该类的成员变量如下（访问权限均为private）：

|  |  |  |
| --- | --- | --- |
| **变量名** | **含义** | **数据类型** |
| id | 帐号 | String |
| owner | 账户持有人姓名 | String |
| balance | 余额 | double |

该类的成员方法如下（访问权限均为public）：

|  |  |  |  |
| --- | --- | --- | --- |
| **方法名** | **返回类型** | **参数** | **说明** |
| 构造方法 |  | 无 | 构造一个账户实例，将id，owner设为null，balance设为0.00 |
| 构造方法 |  | String id, owner, double amount | 构造一个账户实例，用参数设置成员变量id，owner，balance的值 |
| setID | void | String id | 用参数设置成员变量id的值。 |
| setOwner | void | String owner | 用参数设置成员变量owner的值。 |
| deposit | void | String id, double amount | 将金额amount存入id帐户，如果id帐号为null，则打印信息为“帐号未知！” 否则打印“成功存款×××元,当前余额为×××元！”。 |
| withdraw | void | String id, double amount | 从id帐户支取金额amount，如果帐号为null或者余额小于amount，则打印信息分别为“帐号未知！”和“余额不足！”，否则打印“成功取款×××元，当前余额为×××元！” |
| queryBalance | void | 无 | 打印id，owner，balance。 |

|  |
| --- |
| * **问题分析** * 题目要求编写一个银行账户类Account，并在主类中对Account类进行测试。测试步骤为：先存入随机金额，再取款随机金额，最后查询账户信息。 |
| * **代码实现**   import java.util.Random; public class Account {  private String \_id;  private String \_owner;  private double \_balance;   public Account() {  \_id = null;  \_owner = null;  \_balance = 0.00;  }  public Account(String id, String owner, double balance) {  \_id = id;  \_owner = owner;  \_balance = balance;  }  public void setID(String id) {  \_id = id;  }  public void setOwner(String owner) {  \_owner = owner;  }  public void deposit(String id, double amount) {  if (id == null) {  System.*out*.println("账号未知！");  }  else {  \_balance += amount;  System.*out*.println("成功存款" + amount + "元，当前余额为" + \_balance + "元！");  }  }  public void withdraw(String id, double amount) {  if (id ==null || \_balance < amount) {  System.*out*.println("余额不足！");  }  else {  \_balance -= amount;  System.*out*.println("成功取款" + amount + "元，当前余额为" + \_balance + "元！");  }  }  public void queryBalance() {  System.*out*.println("id:" + \_id);  System.*out*.println("owner:" + \_owner);  System.*out*.println("balance:" + \_balance);  }   public static void main(String[] args) {  Account ac1 = new Account("202103151422", "ahwei", 1000000);  Random generate = new Random(20021113);  int money1 = generate.nextInt(1000000);  int money2 = generate.nextInt(1000000);  ac1.deposit("202103151422", money1);  ac1.withdraw("202103151422", money2);  ac1.queryBalance();   Account ac2 = new Account();  ac2.setID("202103151511");  ac2.setOwner("brothor\_cat");  ac2.deposit("202103151511", money1);  ac2.withdraw("202103151511", money2);  ac2.queryBalance();  } } |
| * **运行结果** |
| * **思考及总结**   **在主类的main方法中，创建一个Account对象，调用存款、取款和查询账户信息的方法，将结果输出。** |

**注：**

* **独立完成！切勿抄袭！**