**《Java程序设计》上机作业3**

|  |  |
| --- | --- |
| **学号：** | **202103151422** |
| **姓名：** | **温家伟** |
| **班级：** | **大数据分析** |
| **日期：** | **2023.10.31** |

**1. 根据要求编写程序。**

创建一个名为Rectangle的类，来表示一个使用宽度和高度作为属性的矩形。矩形的宽度和高度由构造方法来确定（要求长和高的范围为10~50）。为Rectangle类创建下列方法：

① getArea返回矩形的面积;

② getPerimeter返回矩形的周长;

③ drawRect使用星号(\*)作为描绘字符画出该矩形的形状（画出的矩形为空心，边长为宽度和高度去除小数点之后的整数）。

在另一个类TestRectangle中编写main方法来测试Rectangle类。

|  |
| --- |
| * **问题分析**   我们需要创建一个名为Rectangle的类来表示矩形，该类具有宽度和高度作为属性，并且需要实现计算面积、周长以及绘制矩形的方法。 |
| * **代码实现**   package test3;  class Rectangle {  private double width = 10.0f;  private double height = 10.0f;  public Rectangle(double w, double h) {  if ((w < 10 || w > 50) || (h < 10 || h > 50)) {  System.*out*.println("长和高的范围为10~50!!!");  }  else {  width = w;  height = h;  }  }  public double getArea() {  return width \* height;  }  public double getPerimeter() {  return 2 \* (width + height);  }  public void drawRect() {  int w = (int)width;  int h = (int) height;  for (int i = 0; i < w; i++) {  System.*out*.print("\*");  }  System.*out*.println("");  for (int i = 0; i < h - 2; i++) {  System.*out*.print("\*");  for (int j = 0; j < w - 2; j++) {  System.*out*.print(" ");  }  System.*out*.print("\*");  System.*out*.println("");  }  for (int i = 0; i < w; i++) {  System.*out*.print("\*");  }  System.*out*.println("");  } } public class TestRectangle {  public static void main(String[] args) {  Rectangle rect = new Rectangle(20, 10);  System.*out*.println("面积是：");  System.*out*.println(rect.getArea());  System.*out*.println("周长是：");  System.*out*.println(rect.getPerimeter());  System.*out*.println("形状是：");  rect.drawRect();  } } |
| * **运行结果** |
| * **思考及总结**   首先，我们需要创建一个Rectangle类，并在构造方法中接收宽度和高度的参数。我们可以添加验证逻辑来确保宽度和高度在10到50之间。  接下来，我们可以实现getArea方法，它将返回矩形的面积，即宽度乘以高度。  然后，我们可以实现getPerimeter方法，它将返回矩形的周长，即将宽度和高度相加后乘以2。  最后，我们可以实现drawRect方法，使用星号(\*)来绘制矩形的形状。我们可以根据矩形的宽度和高度去除小数点之后的整数值来确定矩形的大小。 |

**2.** Define an interface, called **VolumeArea**, within which there’re:

* a static and final variable PI (with the value 3.14159), and
* two abstract methods volume(double radius) and area(double radius).

Write another class **MyCircle** which implements the interface **VolumeArea**. Overriding the two methods:

* volume(double radius) to return a value zero (as the volume of a circle is 0), and
* area(double radius) to return the area of the circle.

Write the third class **MySphere**, also implements **VolumeArea**. Overriding the two methods:

* volume(double radius) to calculate and return the volume of the sphere (using (4π\*r\*r\*r)/3);
* area(double radius) to calculate and return the surface area of the sphere (using 4π\*r\*r).

Write a main class with the main method to test the methods in **MyCircle** and **MySphere**, set the radius of the circle and sphere both to 3.5. And display the results.

|  |
| --- |
| * **问题分析** * 这个问题需要定义一个接口VolumeArea，其中包含一个静态和常量变量PI（值为3.14159）以及两个抽象方法volume(double radius)和area(double radius)。然后，需要编写一个MyCircle类来实现VolumeArea接口，并重写其中的两个方法：volume(double radius)返回0（因为圆的体积为0），area(double radius)返回圆的面积。接着，需要编写第三个类MySphere，也实现了VolumeArea接口，并重写其中的两个方法：volume(double radius)用于计算和返回球体的体积（使用公式(4πrrr)/3)），area(double radius)用于计算和返回球体的表面积（使用公式4πr\*r）。最后，需要编写一个主类，其中包含main方法来测试MyCircle和MySphere中的方法，将圆和球的半径都设置为3.5，并显示结果。 |
| * **代码实现**   package test5;  interface VolumeArea {  static final double *PI* = Math.*PI*;  abstract double volume(double radius);  abstract double area(double radius); }  class MyCircle implements VolumeArea {  public double volume(double radius) {  return 0;  }   public double area(double radius) {  return *PI* \* radius \* radius;  } }  class MySphere implements VolumeArea {  public double volume(double radius) {  return (4 \* *PI* \* radius \* radius \* radius) / 3;  }   public double area(double radius) {  return 4 \* *PI* \* radius \* radius;  } }  public class Main {  public static void main(String[] args) {  double radius = 3.5;   MyCircle circle = new MyCircle();  System.*out*.println("圆面积: " + circle.area(radius));   MySphere sphere = new MySphere();  System.*out*.println("球体积: " + sphere.volume(radius));  System.*out*.println("球表面积: " + sphere.area(radius));  } } |
| * **运行结果** |
| * **思考及总结**  1. 首先，我们需要创建一个名为VolumeArea的接口，在接口中声明一个静态和常量变量PI，并定义两个抽象方法volume(double radius)和area(double radius)。 2. 接着，我们需要编写一个MyCircle类，它实现了VolumeArea接口，并重写其中的两个方法：    * volume(double radius)方法返回0，因为圆的体积为0。    * area(double radius)方法计算并返回圆的面积，可以使用公式Area = π \* r \* r。 3. 然后，我们需要编写一个MySphere类，也实现了VolumeArea接口，并重写其中的两个方法：    * volume(double radius)方法计算并返回球体的体积，可以使用公式Volume = (4/3) \* π \* r \* r \* r。    * area(double radius)方法计算并返回球体的表面积，可以使用公式Surface Area = 4 \* π \* r \* r。 4. 最后，我们需要编写一个主类，其中包含main方法，用于测试MyCircle和MySphere类中的方法。在main方法中，我们可以创建MyCircle和MySphere的实例，将半径设置为3.5，并调用它们的方法来显示结果。 |

**3.** **根据类的继承和多态编写程序实现披萨制作。**

(1) 定义披萨类。属性：名称、价格、大小；方法：显示披萨的属性信息；

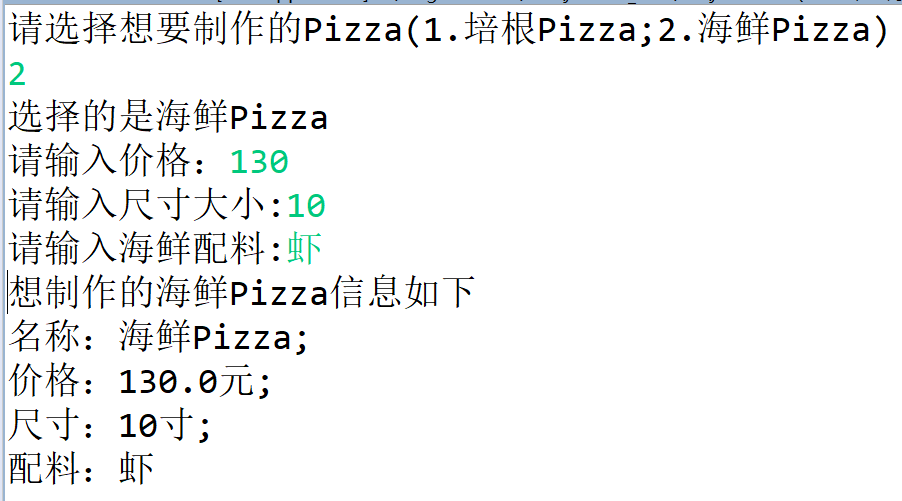
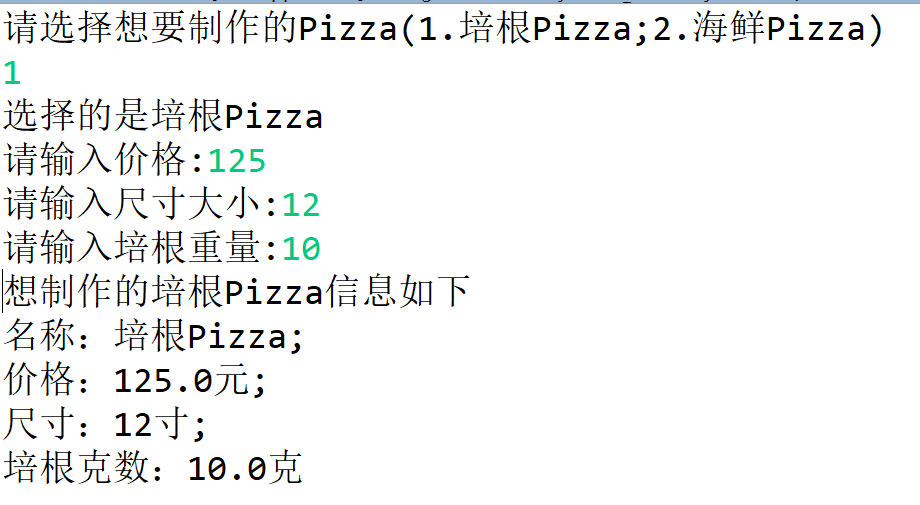
(2) 定义培根披萨和海鲜披萨，继承自披萨类。

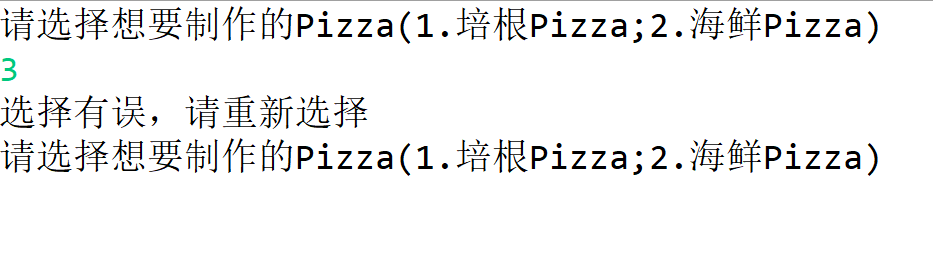
培根披萨的属性：名称、价格、大小、培根克数；方法：显示培根披萨的属性信息；

海鲜披萨的属性：名称、价格、大小、配料信息；方法：显示海鲜披萨的属性信息；

(3) 测试类，主要是根据用户输入的信息制作披萨。

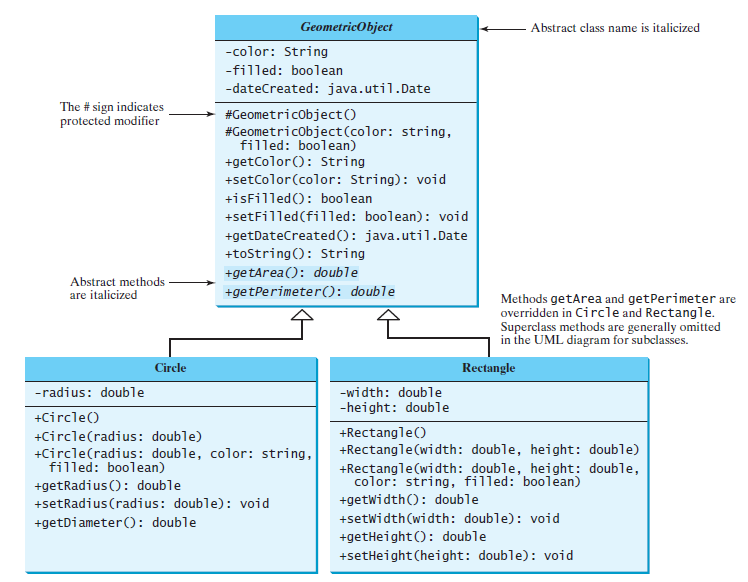
接收用户输入的信息，选择需要制作的披萨。可供选择的披萨有：培根披萨和海鲜披萨。如果选择制作培根披萨，需要用户从键盘输入培根披萨的相关属性，并输出培根披萨的相关属性；如果选择制作海鲜披萨，需要用户从键盘输入海鲜披萨的相关属性，并输出海鲜披萨的相关属性。如果选择其它披萨，则需显示“选择有误，请重新选择”，回到重新选择提示。运行效果下图所示：





|  |
| --- |
| * **问题分析**   我们需要定义一个披萨类，包括属性名称、价格和大小，并具有方法来显示披萨的属性信息。然后我们需要创建两个子类：培根披萨和海鲜披萨，继承自披萨类，并且添加一些新的属性和方法。最后，我们需要编写一个测试类，接收用户输入的信息来制作披萨，并显示相应的属性信息。 |
| * **代码实现**   Pizza类的实现：    海鲜Pizza类的实现    培根Pizza类的实现：    测试类的实现： |
| * **运行结果** |
| * **思考及总结**  1. 首先，我们需要定义一个Pizza类，该类具有名称、价格和大小属性，并且提供一个打印Pizza信息的方法。 2. 接下来，我们可以创建BaconPizza和SeafoodPizza类，这两个类继承了Pizza类并添加了新的属性和方法，例如BaconPizza类有一个baconWeight属性和一个打印BaconPizza信息的方法，而SeafoodPizza类有一个toppingIngredient属性和一个打印SeafoodPizza信息的方法。 3. 最后，我们可以创建一个TestPizza类，从用户那里接收输入信息，选择需要制作的披萨类型。如果用户选择制作BaconPizza，则需要他/她输入BaconPizza的相关属性，并显示输出BaconPizza信息。如果用户选择制作SeafoodPizza，则需要他/她输入SeafoodPizza的相关属性，并显示输出SeafoodPizza信息。如果用户选择其他披萨类型，则需要显示“选择有误，请重新选择”，并回到重新选择提示。 |

**4. 请根据以下UML图，编写相关的三个类**



定义一个主类，main方法中对上图中设计的各个方法进行测试。

TIPs: ’+’符号表示public修饰的属性/方法；’-‘表示private修饰；’#’表示protected修饰；斜体表示abstract修饰的类/方法。

|  |
| --- |
| * **问题分析**   按照要求写出各个类的成员和方法，注意抽象方法的实现。 |
| * **代码实现**   GeometricObject类：    Circle类：    Rectangle类：    测试类： |
| * **运行结果** |
| * **思考及总结**   抽象方法要在子类中实现。 |

**5. 按如下要求，编写应用程序：**

编写一个抽象类Animal，其成员变量有name，age，weight表示动物名、年龄和重量。方法有showInfo( )、move( )和eat( )，其中后面两个方法是抽象方法；

编写一个类Bird继承Animal，实现相应的方法，通过构造方法给name，age，weight分别赋值，showInfo( )打印鸟名、年龄和重量，move( )方法打印鸟的运动方式，eat( )打印鸟喜欢吃的食物；

编写一个类Dog继承Animal，实现相应的方法，通过构造方法给name，age，weight分别赋值，showInfo( )打印狗名、年龄和重量，move( )方法打印狗的运动方式，eat( )打印狗喜欢吃的食物；

最后编写测试类TestAnimal，用Animal类型的变量，调用Bird和Dog对象的三个方法。

要求：将Animal、Bird和Dog三个类定义在包中（包名为你的名字或姓名缩写），而测试类TestAnimal在缺省包中。

|  |
| --- |
| * **问题分析**   我们需要定义一个抽象类Animal，包括成员变量name、age和weight，以及方法showInfo()、move()和eat()。然后创建Bird类和Dog类来继承Animal类，并实现相应的方法。最后编写一个测试类TestAnimal，在该类中使用Animal类型的变量调用Bird和Dog对象的方法。 |
| * **代码实现**   Animal类：    鸟类：    狗类：    测试类：    目录结构： |
| * **运行结果** |
| * **思考及总结**   首先，我们需要定义一个抽象类Animal，其中包括成员变量name、age和weight，以及方法showInfo()、move()和eat()。其中，move()和eat()方法是抽象方法，需要在子类中进行实现。  接下来，我们可以创建Bird类继承Animal类，并实现move()和eat()方法。在构造方法中给name、age和weight赋值，并在showInfo()方法中打印鸟的信息。  然后，我们可以创建Dog类继承Animal类，并实现move()和eat()方法。在构造方法中给name、age和weight赋值，并在showInfo()方法中打印狗的信息。  最后，我们可以创建一个测试类TestAnimal，在该类中使用Animal类型的变量分别实例化Bird和Dog对象，并通过调用它们的方法来展示鸟和狗的信息。 |

**6.** (*Math: The* **Complex** *class*) A complex number is a number of the form where *a* and *b* are real numbers and *i* is .The numbers **a** and **b** are known as the real part and imaginary part of the complex number, respectively. You can perform addition, subtraction, multiplication, and division for complex numbers using the following formula:

文本, 信件

描述已自动生成

You can also obtain the absolute value for a complex number using the following formula:



Design a class named **Complex** for representing complex numbers and the methods **add**, **subtract**, **multiply**, **divide**, and **abs** for performing complex-number operations, and override the **toString** method for returning a string representation for a complex number. The **toString** method returns **a + bi** as a string. If **b** is **0**, it simply returns **a**.

Provide three constructors **Complex(a, b)**, **Complex(a)**, and **Complex()**. **Complex()** creates a **Complex** object for number **0** and **Complex(a)** creates a **Complex** object with **0** for **b**. Also provide the **getRealPart()** and **getImaginaryPart()** methods for returning the real and imaginary part of the complex number, respectively.

Write a test program that prompts the user to enter two complex numbers and display the result of their addition, subtraction, multiplication, and division. Here is a sample run:

|  |
| --- |
| Enter the first complex number: 3.5 5.5  Enter the second complex number: -3.5 1  3.5 + 5.5i + -3.5 + 1.0i = 0.0 + 6.5i  3.5 + 5.5i - -3.5 + 1.0i = 7.0 + 4.5i  3.5 + 5.5i \* -3.5 + 1.0i = -17.75 + -15.75i  3.5 + 5.5i / -3.5 + 1.0i = -0.5094 + -1.7i  |3.5 + 5.5i| = 6.519202405202649 |

|  |
| --- |
| * **问题分析**   用户需要一个Java程序，能够进行复数的四则运算，并输出结果。程序需要接受用户输入的两个复数，然后对这两个复数进行加法、减法、乘法和除法运算，并将结果输出。 |
| * **代码实现**   package test5;  import java.util.Scanner;  public class Complex {  private double real;  private double imag;  public Complex(double a, double b) {  real = a;  imag = b;  }   public Complex(double a) {  real = a;  imag = 0;  }   public Complex() {  real = 0;  imag = 0;  }   public double getRealPart() {  return real;  }   public double getImaginaryPart() {  return imag;  }   public Complex add(Complex complex) {  double r = real + complex.getRealPart();  double i = imag + complex.getImaginaryPart();  return new Complex(r, i);  }   public Complex subtract(Complex complex) {  double r = real - complex.getRealPart();  double i = imag - complex.getImaginaryPart();  return new Complex(r, i);  }   public Complex multiply(Complex complex) {  double r = real \* complex.getRealPart() - imag \* complex.getImaginaryPart();  double i = real \* complex.getImaginaryPart() + imag \* complex.getRealPart();  return new Complex(r, i);  }   public Complex divide(Complex complex) {  double temp = Math.*pow*(complex.getRealPart(), 2) + Math.*pow*(complex.getImaginaryPart(), 2);  double r = (real \* complex.getRealPart() + imag \* complex.getImaginaryPart()) / temp;  double i = (imag \* complex.getRealPart() - real \* complex.getImaginaryPart()) / temp;  return new Complex(r, i);  }   public double abs() {  return Math.*sqrt*(Math.*pow*(real, 2) + Math.*pow*(imag, 2));  }   public String toString() {  if (imag == 0) {  return "" + real;  }  else {  return real + "+" + imag + "i";  }  }   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Enter the first complex number: ");  double a1 = scanner.nextDouble();  double b1 = scanner.nextDouble();  Complex complex1 = new Complex(a1, b1);   System.*out*.print("Enter the second complex number: ");  double a2 = scanner.nextDouble();  double b2 = scanner.nextDouble();  Complex complex2 = new Complex(a2, b2);   System.*out*.println("(" + complex1 + ") + (" + complex2 + ") = " + complex1.add(complex2));  System.*out*.println("(" + complex1 + ") - (" + complex2 + ") = " + complex1.subtract(complex2));  System.*out*.println("(" + complex1 + ") \* (" + complex2 + ") = " + complex1.multiply(complex2));  System.*out*.println("(" + complex1 + ") / (" + complex2 + ") = " + complex1.divide(complex2));  System.*out*.println("|" + complex1 + "| = " + complex1.abs());  } } |
| * **运行结果** |
| * **思考及总结**  1. 首先，我们需要定义一个Complex类来表示复数。该类应该包含实部（real）和虚部（imag）两个私有属性，并提供构造方法和相关的获取方法（getRealPart()和getImaginaryPart()）。 2. 在Complex类中，我们需要实现复数的加法、减法、乘法和除法运算。加法和减法可以通过将实部和虚部分别相加或相减来完成，乘法和除法则需要按照对应的公式进行计算。 3. 我们还可以为Complex类添加计算复数绝对值的方法（abs()），利用勾股定理计算模长。 4. 在主方法中，我们需要使用Scanner类获取用户输入的两个复数的实部和虚部。 5. 创建两个Complex对象，分别代表用户输入的两个复数。 6. 调用Complex类中的方法，对这两个复数进行加法、减法、乘法和除法运算，并将结果输出到控制台。 |