多态作业

一、 选择题

1. **关于 Java 中的多态，以下说法不正确的为（ B ）。（选择一项）**

A 多态不仅可以减少代码量，还可以提高代码的可扩展性和可维护性
B. 把子类转换为父类，称为向下转型，自动进行类型转换
C. 多态是指同一个实现接口，使用不同的实例而执行不同的操作
D. 继承是多态的基础，没有继承就没有多态

2. **编译运行如下 Java 代码，输出结果是（ D ）。（选择一项）**

```
class Base {
    public void method(){
        System.out.print ("Base method");
    }
}
class Child extends Base{
    public void methodB(){
        System.out.print ("Child methodB");
    }
}
class Sample {
    public static void main(String[] args) {
        Base base= new Child();
        base.methodB();
    }
}
```

A Base method
B. Child methodB
C. Base method
Child methodB
D. 编译错误

3. **在 Java 中，关于引用数据类型的类型转换说法正确的是（ AB ）。（选择二项）**

A 引用数据类型的类型转换有向上转型和向下转型
B. 向下转型，必须转换成其真实子类型，而不能随意转换
C. 向下转型是自动进行的，也称隐式转换
D. 向上转型可以使用instanceof操作符来判断转型的合法性

4. **给定如下 Java 程序，Test 类中的四个输出语句输出结果依次是（ C ）。（选择一项）**

```
class Person {
```

```
    String name="person";
    public void shout(){
        System.out.println(name);
    }
}
class Student extends Person{
    String name="student";
    String school="school";
}
public class Test {
    public static void main(String[ ] args) {
        Person p=new Student();
        System.out.println(p instanceof Student);
        System.out.println(p instanceof Person);
        System.out.println(p instanceof Object);;
        System.out.println(p instanceof System);
    }
}
```

A true,false,true,false

B. false,true,false,false

C. true,true,true,编译错误

D. true,true,false,编译错误

二、 判断题

1. 将子类对象赋给父类引用变量，称为向下转型，将无法访问子类特有的方法。(F)
2. 继承是多态的基础，没有继承就没有多态。(T)

三、 简答题

1. 多态的含义和作用
2. 举一个例子说明多态
3. 多态的三个必要条件
4. 向上转型和向下转型

四、 编码题

1. 编写程序实现乐手弹奏乐器。乐手可以弹奏不同的乐器从而发出不同的声音。可以弹奏的乐器包括二胡、钢琴和琵琶。

   实现思路及关键代码

   1) 定义乐器类 Instrument，包括方法 makeSound()
   2) 定义乐器类的子类：二胡 Erhu、钢琴 Piano 和小提琴 Violin
   3) 定义乐手类 Musician，可以弹奏各种乐器 play(Instrument i)
   4) 定义测试类，给乐手不同的乐器让他弹奏

2. 编写程序实现比萨制作。需求说明编写程序，接收用户输入的信息，选择需要制作的比萨。可供选择的比萨有：培根比萨和海鲜比萨。

   实现思路及关键代码

1　多态的含义和作用

多态是一个对象具有多种状态的行为

2　举一个例子说明多态

Dog d = new Dog();

Animal animal = new Animal;

animal = d;

animal.speek();动物就会变成了狗

3　多态的三个必要条件

要有继承

要有重写

父类的引用指向子类的对象

4　向上转型和向下转型

父类的引用指向了子类的对象，该写法是对象的向上转型

子类的引用指向了父类的对象，该写法是对象的向下转型；向下转型不是自动的，需要　强制类型转换符进行转换

1

```java
public class Erhu extends Instrument {
    1个用法  新 *
    @Override
    protected void makeSound() { System.out.println("二胡声"); }
}
```

```java
public abstract class Instrument {
    1个用法  3 个实现  新 *
    protected abstract void makeSound();
}
```

```java
public class Musician {
    3 个用法  新 *
    protected void play(Instrument i) { i.makeSound(); }
}
```

```java
public class Piano extends Instrument{
    1个用法  新 *
    @Override
    protected void makeSound() { System.out.println("钢琴声"); }
}
```

```java
public class Violin extends Instrument {
    1个用法  新 *
    @Override
    protected void makeSound() { System.out.println("小提琴声"); }
}
```

```java
public class Test {
    新 *
    public static void main(String[] args) {
        Musician musician = new Musician();
        Instrument erhu = new Erhu();
        Piano piano = new Piano();
        Violin violin = new Violin();
        musician.play(erhu);
        musician.play(piano);
        musician.play(violin);
    }
}
```

2

```java
public class BaconPizza extends Pisa{
    2 个用法
    public Integer gram;
    1 个用法
    private String name = "培根披萨";

    1 个用法  新 *
    public Integer getGram() { return gram; }

    1 个用法  新 *
    public void setGram(Integer gram) { this.gram = gram; }

    新 *
    @Override
    public void show() {
        System.out.println("名称: " + this.name);
        System.out.println("价格: " + this.getPrice());
        System.out.println("大小: " + this.getSize());
        System.out.println("培根克数: " + this.getGram());
    }
}
```

```java
public class Factory {
    1 个用法  新 *
    public Pisa pisaClassification(int n){
        if (n == 1){
            return new BaconPizza();
        }
        if (n == 2){
            return new SeafoodPizza();
        }
        return null;
    }
}
```

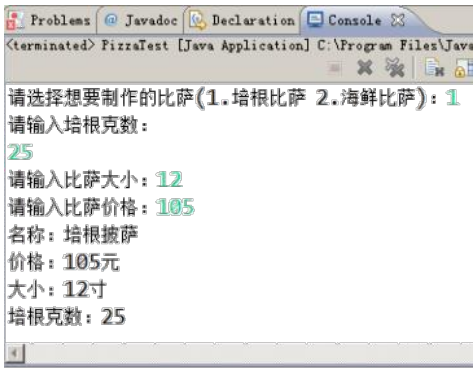
```java
public class Test {
    新 *
    public static void main(String[] args) throws ClassNotFoundException {
        System.out.print("请选择想要制造的比萨(1. 培根披萨 2. 海鲜披萨): ");
        Scanner scanner = new Scanner(System.in);
        int input = scanner.nextInt();
        Factory factory = new Factory();
        Pisa p1 = factory.pisaClassification(input);

        System.out.println("请输入培根克数: ");
        int gram = scanner.nextInt();
        BaconPizza b = (BaconPizza) p1;
        b.setGram(gram);

        System.out.println("请输入配料信息: ");
        String ingredient = scanner.next();
        String ingredient1 = scanner.next();
        String ingredient2 = scanner.next();
        SeafoodPizza s = (SeafoodPizza) p1;
        s.setFoodIngredient(ingredient);
        s.setFoodIngredient(ingredient1);
        s.setFoodIngredient(ingredient2);
        System.out.print("请输入比萨大小: ");
        int size = scanner.nextInt();
        p1.setSize(size);
        System.out.print("请输入比萨价格: ");
        int price = scanner.nextInt();
        p1.setPrice(price);
        p1.show();
```

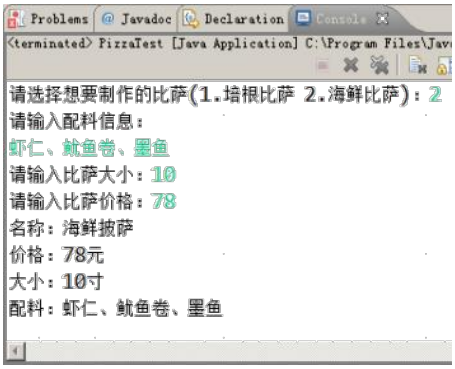
```java
public abstract class Pisa {
    2 个用法
    private String name;
    2 个用法
    private Integer price;
    2 个用法
    private Integer size;


    新 *
    public String getName() { return name; }

    新 *
    public void setName(String name) {
        this.name = name;
    }

    2 个用法  新 *
    public Integer getPrice() { return price; }

    1 个用法  新 *
    public void setPrice(Integer price) { this.price = price; }

    2 个用法  新 *
    public Integer getSize() { return size; }

    1 个用法  新 *
    public void setSize(Integer size) { this.size = size; }

    2 个实现  新 *
    public abstract void show();
}
```

```java
public class SeafoodPizza extends Pisa{
    5 个用法
    private List<String> foodIngredient = new ArrayList<>();
    1 个用法
    private String name = "海鲜披萨";

    0 个用法  新 *
    public List<String> getFoodIngredient() { return foodIngredient; }

    3 个用法  新 *
    public void setFoodIngredient(String ingredient) { this.foodIngredient.add(ingredient); }

    新 *
    @Override
    public void show() {
        System.out.println("名称: " + this.name);
        System.out.println("价格: " + this.getPrice());
        System.out.println("大小: " + this.getSize());
        System.out.print("配料: " + this.foodIngredient.get(0));
        System.out.print("、");
        System.out.print(this.foodIngredient.get(1));
        System.out.print("、");
        System.out.print(this.foodIngredient.get(2));
    }
}
```

1) 分析培根比萨和海鲜比萨
2) 定义比萨类
   a) 属性：名称、价格、大小
   b) 方法：展示
3) 定义培根比萨和海鲜比萨继承自比萨类
4) 定义比萨工厂类，根据输入信息产生具体的比萨对象

程序运行结果如图所示





五、 可选题

1. 编写程序实现软料购买：编写程序，接收用户输入的信息，选择购买的饮料。可供选择的饮料有：咖啡、矿泉水和可乐。其中，购买咖啡时可以选择：加糖、加奶还是什么都不加。购买可乐时可以选择：买可口可乐还是百事可乐。

   程序运行效果如图所示。