# 第5章作业（下）

（共100分）

一、单选题，并说明理由（对的为什么对，错的为什么错，每个选项都要评论）（15分，每题5分，其中选对2分，解释占3分）

1. 在Java中，一个类可同时定义许多同名的方法，这些方法的形式参数个数、类型或顺序各不相同，传回的值也可以不相同。这种面向对象程序的特性称为（ C ）。

A. 隐藏 B. 覆盖 C. 过载 D. Java不支持此特性

a隐藏通常用于静态成员，包括静态方法和静态变量，它不实现多态性

b覆盖用于实例方法，要求方法签名相同，实现多态性

2．下列关于修饰符使用的说法，错误的是（ ）。

A. abstract不能与final并列修饰同一个类

B. abstract类中可以有private的成员

C. abstract方法不能在非abstract类中定义

D. 接口中可以有private的成员。

a抽象类表示类的抽象性，可以被子类继承和扩展，而最终类表示不可被继承。

b在抽象类中可以有private的成员，包括私有字段（成员变量）和私有方法。

c抽象方法必须在抽象类中声明但不能有具体实现。

3．构造B类的一个实例b，该类实现了A接口，下列（ ）代码是正确的。

A. A b = new B( ); B. A b = new A( );

C. B b = new A( ); D. B b = new Object( );

a因为接口A被类B实现，因此你可以使用接口类型A来引用类B的一个实例。

b接口不能直接实例化

c类A不能直接赋值给类B的引用变量，除非A是B的子类。

dObject 是一个Java中所有类的根类，但它并不是 B 的子类，因此无法用 B 类型的引用来引用 Object 类的对象

二、程序阅读题，写出程序运行的结果，并进行解释（每题15分，共45分；其中每题运行结果10分，解释5分）

1、

class ClassA {

int result = -1;

void fun(int x, int y){

result += x+y;

}

public ClassA() {

System.out.println("Constructing Class A");

}

{

System.out.println("Class A");

}

}

public class ClassB extends ClassA {

void fun(double x, double y) {

result = (int) (y-x);

}

void show(){

System.out.println(result);

}

public ClassB() {

System.out.println("Constructing Class B");

fun(10,20);

show();

}

{

result = 1;

System.out.println("Class B");

}

public static void main(String[] args) {

ClassB Obj = new ClassB();

}

}

Class A

Constructing Class A

Class B

Constructing Class B

31

先执行父类的实例变量初始化和父类的实例代码块，打印Class A ，再执行父类的构造方法，打印Constructing Class A，再执行子类的实例变量初始化和子类的实例代码块，父类的result被继承且被赋值为1，打印Class B，再执行子类的构造方法，打印Constructing Class B

，由于10和20是int型，故执行父类fun方法，result变成31并打印

2.

class HelloA {

public HelloA() {

System.out.println("HelloA");

}

{

System.out.println("I'm A class");

}

static {

System.out.println("static A");

}

}

public class AnalysisTest extends HelloA {

public AnalysisTest() {

System.out.println("HelloB");

}

{

System.out.println("I'm B class");

}

static {

System.out.println("static B");

}

public static void main(String[] args) {

new AnalysisTest();

}

}

static A

static B

I'm A class

HelloA

I'm B class

HelloB

1.先执行父类的静态变量初始化和静态代码块（顺序只跟它们在代码中出现的先后有关）

2执行子类的静态变量初始化和静态代码块（顺序只跟它们在代码中出现的先后有关）

3执行父类的实例变量初始化和父类的实例代码块（顺序只跟它们在代码中出现的先后有关）

4执行父类的构造方法

5执行子类的实例变量初始化和子类的实例代码块（顺序只跟它们在代码中出现的先后有关）

6执行子类的构造方法

3、

public class Test {

public static void main(String[] args) {

Animal a;

Animal.whoAmI();

Cat c = new Cat();

Dog d = new Dog();

a = c;

a.whoAmI();

a.eat();

a = d;

a.whoAmI();

a.eat();

c.whoAmI();

d.whoAmI();

}

}

abstract class Animal {

abstract void eat();

static void whoAmI() {

System.out.println("动物");

}

}

class Cat extends Animal {

public void eat() {

System.out.println("吃鱼");

}

static void whoAmI() {

System.out.println("猫");

}

}

class Dog extends Animal {

public void eat() {

System.out.println("吃骨头");

}

static void whoAmI() {

System.out.println("狗");

}

}

动物

动物

吃鱼

动物

吃骨头

猫

狗

1. 在 `main` 方法中，首先声明了一个 `Animal` 类型的引用变量 `a`。

2. 调用 `Animal.whoAmI()` 静态方法，该方法输出 "动物"，因为静态方法不受多态性的影响，它会根据引用类型来确定要调用的方法。

3. 创建了一个 `Cat` 类的实例 `c` 和一个 `Dog` 类的实例 `d`。

4. 将 `c` 赋值给 `a`，此时 `a` 引用的是一个 `Cat` 类的实例。然后调用 `a.whoAmI()`，由于静态方法不受多态性的影响，会调用 `animal`类中的 `whoAmI()`方法，输出 "动物"。

5. 调用 `a.eat()`，由于 `a` 引用的是 `Cat` 类的实例，会调用 `Cat` 类中的 `eat()` 方法，输出 "吃鱼"。

6. 将 `d` 赋值给 `a`，此时 `a` 引用的是一个 `Dog` 类的实例。然后再次调用 `a.whoAmI()`，同理由于是静态方法， 会输出"动物"。

7. 调用 `a.eat()`，由于 `a` 引用的是 `Dog` 类的实例，会调用 `Dog` 类中的 `eat()` 方法，输出 "吃骨头"。

8. 分别调用 `c.whoAmI()` 和 `d.whoAmI()`，它们分别输出 "猫" 和 "狗"，因为这两个方法是被类直接调用的，不受多态性的影响。

三、编程题（每题20分，共 40分；题干中若要求输出结果，不仅要有代码，还要有运行结果）

1、有如下2个接口：

interface Ia {

int max(int[] a);

}

interface Ib {

int min(int[] a);

}

其中max方法为返回数组中元素的最大值，min方法为返回数组中元素的最小值。

要求：设计一个非抽象类Test2，同时实现这两个接口，并在Test2类的main方法中用如下考试的分数数据{ 88, 89, 82, 90, 98 }初始化一个数组，并分别使用max和min方法求出最高分和最低分，并打印输出。

interface Ia {

  int max(int[] a);

  }

interface Ib {

  int min(int[] a);

  }

class Test2 implements Ia, Ib {

  public int max(int[] a) {

      int maxScore = Integer.MIN\_VALUE;

      for (int score : a) {

          if (score > maxScore) {

              maxScore = score;

          }

      }

      return maxScore;

  }

  public int min(int[] a) {

      int minScore = Integer.MAX\_VALUE;

      for (int score : a) {

          if (score < minScore) {

              minScore = score;

          }

      }

      return minScore;

  }

  public static void main(String[] args) {

      int[] scores = { 88, 89, 82, 90, 98 };

      Test2 test = new Test2();

      int maxScore = test.max(scores);

      int minScore = test.min(scores);

      System.out.println("最高分: " + maxScore);

      System.out.println("最低分: " + minScore);

  }

}



2、求正方形的面积和缩放形状。

(1) 先创建一个接口IShape ，接口中有2个抽象方法：

double area( ); // 计算面积

square zoom(double factor); // zoom是缩放方法，返回一个新的square对象，该对象的面积等于原正方形面积乘以缩放因子。其中factor是缩放因子，factor必须是正数，否则打印出错信息。

(2) 再定义一个正方形类square，实现IShape接口。另外还需包括的成员有：

a) 一个私有数据成员，表示正方形的边长；

b) 一个构造方法，初始化边长，要求边长必须为正数，否则打印出错信息；

c) 覆盖公共的toString方法，输出格式为：“正方形的边长:\*.\*\*; 正方形的面积:\*.\*\*”，保留小数后两位；

(3) 再定义一个主类SquareTest。在此类中，先创建一个变长为10的square对象，计算该正方形面积，并将边长和面积输出到console端；再根据缩放因子0.25生成一个新的square对象，计算该正方形面积和新的边长，并将边长和面积输出到console端。

interface IShape {

  double area();

  Square zoom(double factor);

}

class Square implements IShape {

  private double sideLength;

  public Square(double sideLength) {

      if (sideLength <= 0) {

          System.err.println("边长必须为正数");

          System.exit(1);

      }

      this.sideLength = sideLength;

  }

  @Override

  public double area() {

      return sideLength \* sideLength;

  }

  @Override

  public Square zoom(double factor) {

      if (factor <= 0) {

          System.err.println("缩放因子必须为正数");

          System.exit(1);

      }

      double newSideLength = sideLength \* Math.sqrt(factor);

      return new Square(newSideLength);

  }

  @Override

  public String toString() {

      return String.format("正方形的边长: %.2f; 正方形的面积: %.2f", sideLength, area());

  }

}

public class SquareTest {

  public static void main(String[] args) {

      Square square1 = new Square(10);

      System.out.println(square1);

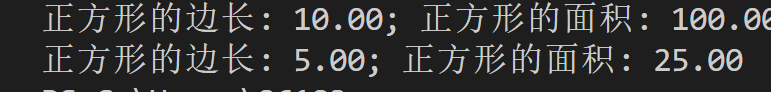
      double scaleFactor = 0.25;

      Square square2 = square1.zoom(scaleFactor);

      System.out.println(square2);

  }

}



---------------------------

提交方式：

1. 上传到坚果云的[此链接中](https://send2me.cn/L49owuPV/R-W9BdHNPgdZhg)（上传时请勿挂vpn或其他梯子，否则可能上传失败）。
2. 文件名严格按照：学号-姓名-第几次作业.docx(或pdf)的方式命名。
3. 每次作业在截止日期之前，若发现提交内容有误，可以再提交一次。但每次作业最多只能提交两次。第二次提交的文件后面加上“（2）”的字样。
4. 作业提交形式：单一的word或pdf文档。不要上传Eclipse、VScode、IDEA等项目文件，不要压缩打包，不要以文件夹形式上传。
5. 截止时间：9月28日晚23:59。