# 第5章

### 选择题

1. A

A选项是成员内部类的通用实例化写法，静态内部类也是一种成员内部类，内部类以static修饰则为静态成员内部类，无static修饰则为对象成员内部类。本题中，静态内部类B中定义的静态方法myvoid也可以直接通过类名调用，而不必实例化B类的对象，写法为：A.B.myvoid()。

1. A

执行顺序是：A类的构造方法 -> B类的构造方法 -> B类的myVoid1方法。

1. B

A类的实例对象c1通过A类的接口实现继承了接口B的实例变量k。

1. AD

Java语言规定类只能单一继承，面向对象中多重继承的特性是通过接口来实现的，类可以实现多个接口。

### 填空题

对象成员内部类、静态内部类、局部内部类、匿名内部类。

### 简答题

1. 类成员的访问控制符可以是public，不写时为default访问级别；类成员的访问控制符包括public、protected、private，不写时为default访问级别，访问级别从大到小依次为：public、protected、default和private。
2. 关键字static可以修饰类的成员变量、成员方法、成员内部类以及代码块。
3. 程序运行结果：

the original data is:-1

now the data is:10

解析：myobj和myref是指向同一个对象的两个引用类型变量，通过myref对该对象内部数据进行修改后，使用myobj获取的仍是同一个对象变更过的数据。

1. 抽象方法以abstract修饰，仅有方法声明，而没有方法体。抽象方法的方法体应该在具体子类中给出实现定义。抽象方法定义了一般类型的行为接口，由具体子类根据各自情况给出不同的实现，利用抽象方法的这一特点可以设计出高内聚、低耦合的功能模块。
2. final可以用来修饰类、变量和方法，分别表示类不能被继承、变量的值不能被改变（常量）、方法不能被覆写（Override）。
3. 接口中可以包括静态变量和抽象方法。接口中各成员的访问控制符全部默认为public。（Java8以后允许在接口里定义默认方法和静态方法）
4. 创建接口使用interface关键字。接口可以有父接口。

public interface MyInterface extends MySuperInterface1, MySuperInterface2 { }

1. 实现接口的类不必覆盖该接口的所有抽象方法，但该类必须声明为抽象类，且不能实例化，只有实现了全部抽象方法的具体类才能够实例化对象。
2. 接口中定义的抽象方法默认访问级别即为public，进行方法覆盖时访问范围可以扩大，但不能缩小。

### 编程题

1. 参考代码如下：

|  |
| --- |
| /\* Shape.java \*/  **public** **class** Shape {  **int** x, y;  **void** draw() {  System.***out***.println("draw a shape");  }  }  /\* Rectangle.java \*/  **public** **class** Rectangle **extends** Shape {  **int** length, width;    **void** draw() {  System.***out***.println("draw a rectangle");  }  }  /\* Circle.java \*/  **public** **class** Circle **extends** Shape {  **int** radius;    **void** draw() {  System.***out***.println("draw a circle");  }  }  /\* Test.java \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  Shape shape1 = **new** Rectangle();  Shape shape2 = **new** Circle();    shape1.x = 1;  shape1.y = 2;  System.***out***.println("shape.x=" + shape1.x + " shape.y=" + shape1.y);    shape1.draw();  shape2.draw();  }  } |

1. 复数的运算法则：

复数加法：(a + bi) + (c + di) = (a + c) + (b + d)i

复数减法：(a + bi) - (c + di) = (a - c) + (b - d)i

复数乘法：(a + bi) \* (c + di) = (ac - bd) + (ad + bc)i

参考代码如下：

|  |
| --- |
| **public** **class** Question4\_2 {  ComplexNumber plus(ComplexNumber cn1, ComplexNumber cn2) {  **return** **new** ComplexNumber(cn1.x + cn2.x, cn1.y + cn2.y);  }  ComplexNumber minus(ComplexNumber cn1, ComplexNumber cn2) {  **return** **new** ComplexNumber(cn1.x - cn2.x, cn1.y - cn2.y);  }  ComplexNumber multiply(ComplexNumber cn1, ComplexNumber cn2) {  **return** **new** ComplexNumber(cn1.x \* cn2.x - cn1.y \* cn2.y, cn1.x \* cn2.y + cn1.y \* cn2.x);  }  **public** **static** **void** main(String[] args) {  Question4\_2 q = **new** Question4\_2();  ComplexNumber n1 = **new** ComplexNumber(1, 2);  ComplexNumber n2 = **new** ComplexNumber(3, 4);  System.***out***.println("复数1：" + n1.output());  System.***out***.println("复数2：" + n2.output());  System.***out***.println("复数加法结果：" + q.plus(n1, n2).output());  System.***out***.println("复数减法结果：" + q.minus(n1, n2).output());  System.***out***.println("复数乘法结果：" + q.multiply(n1, n2).output());  }  }  **class** ComplexNumber {  **int** x;  **int** y;  **public** ComplexNumber(**int** x, **int** y) {  **this**.x = x;  **this**.y = y;  }  **public** String output() {  **return** y >= 0 ? x + "+" + y + "i" : x + "" + y + "i";  }  } |

1. 参考代码如下：

|  |
| --- |
| **public** **class** Box {  **int** length, width, height;    **public** Box(**int** length, **int** width, **int** height) {  **this**.length = length;  **this**.width = width;  **this**.height = height;  }    **public** **int** volumn() {  **return** length \* width \* height;  }    **public** **static** **void** main(String[] args) {  Box box = **new** Box(1, 2, 3);  System.***out***.println("立方体的体积：" + box.volumn());  }  } |

1. 参考代码如下：

|  |
| --- |
| **public** **class** Student {  **private** String studentNumber;  **private** String classNumber;  **private** String name;  **private** String sex;  **private** **int** studentCountOfClass;  **public** String getStudentNumber() {  **return** studentNumber;  }  **public** **void** setStudentNumber(String studentNumber) {  **this**.studentNumber = studentNumber;  }  **public** String getClassNumber() {  **return** classNumber;  }  **public** **void** setClassNumber(String classNumber) {  **this**.classNumber = classNumber;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getSex() {  **return** sex;  }  **public** **void** setSex(String sex) {  **this**.sex = sex;  }  **public** **int** getStudentCountOfClass() {  **return** studentCountOfClass;  }  **public** **void** setStudentCountOfClass(**int** studentCountOfClass) {  **this**.studentCountOfClass = studentCountOfClass;  }  **public** String toString() {  **return** "学生 [学号=" + studentNumber + ", 班号=" + classNumber + ", 姓名=" + name + ", 性别=" + sex + ", 班级人数="  + studentCountOfClass + "]";  }    **public** **static** **void** main(String[] args) {  Student[] students = **new** Student[2];  Student student1 = **new** Student();  student1.setClassNumber("HIT0003108");  student1.setStudentNumber("123456");  student1.setName("张三");  student1.setSex("男");  student1.setStudentCountOfClass(32);  students[0] = student1;  Student student2 = **new** Student();  student2.setClassNumber("HIT0003109");  student2.setStudentNumber("654321");  student2.setName("李四");  student2.setSex("女");  student2.setStudentCountOfClass(27);  students[1] = student2;    System.***out***.println(students[0].toString());  System.***out***.println(students[1].toString());  }  } |

1. 参考代码如下：

|  |
| --- |
| /\* Person.java \*/  **public** **abstract** **class** Person {  **public** **abstract** **double** pay();  }  /\* Teacher.java \*/  **public** **class** Teacher **extends** Person {  **private** **double** baseSalary;  **private** **int** courseTimes;    **public** **double** getBaseSalary() {  **return** baseSalary;  }  **public** **void** setBaseSalary(**double** baseSalary) {  **this**.baseSalary = baseSalary;  }  **public** **int** getCourseTimes() {  **return** courseTimes;  }  **public** **void** setCourseTimes(**int** courseTimes) {  **this**.courseTimes = courseTimes;  }  @Override  **public** **double** pay() {  **return** baseSalary + courseTimes \* 30;  }  }  /\* CollegeStudent.java \*/  **public** **class** CollegeStudent **extends** Person {  **private** **double** scholarship;    **public** **double** getScholarship() {  **return** scholarship;  }  **public** **void** setScholarship(**double** scholarship) {  **this**.scholarship = scholarship;  }  @Override  **public** **double** pay() {  **return** scholarship;  }  }  /\* Test.java \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  Teacher teacher = **new** Teacher();  teacher.setBaseSalary(3000);  teacher.setCourseTimes(100);  Person p1 = teacher;  System.***out***.println(p1.pay());    CollegeStudent student = **new** CollegeStudent();  student.setScholarship(1000);  Person p2 = student;  System.***out***.println(p2.pay());  }  } |

### 简答题

1. 一般的类可以有两种访问级别，当没有任何访问修饰符时为默认default级别，可以被同一个包的其他类所访问，使用public修饰符时可以被任何类所访问。
2. Inner class和成员变量、成员方法处于外部类中的同一层级，访问级别与类的成员一样，可以是public、protected、default（不写）和private。
3. 继承，就是保持已有类的特性而创造新类的过程，被继承的类称为父类（或基类、超类），而继承者称为子类（或派生类），父类与子类是一般——特殊的关系。继承简化了创建类的工作量，增加了代码的可重用性，提供了类的等级结构。
4. Java不支持类之间的多重继承，但支持接口的多重继承，同样实现了面向对象的多重继承特征。虽然类的多重继承便于程序设计人员理解和编码，但增加了编译器和运行环境的负担。