# 一、简述题部分（每题答3-4点）

**1、重载与重写的区别**

重载：

方法的重载是允许一个类的定义中，多个方法使用相同的方法名，方法的名称必须是相同的，但在方法的声明中一定要有彼此不同的成分，返回类型，修饰符可以相同也可以不同

重写：

指子类重写父类的成员方法，子类可以改写父类方法所实现的功能，但子类中重写的方法必须与父类中对应的方法具有相同的返回值、方法名和参数

**2、静态变量和实例变量的区别**

静态变量：

是一个在类的所有实例对象中都能访问的变量；

静态变量只依附于类，与类的实例对象无关；

非private类型的静态变量，可以在该类外直接用类名调用

实例变量：

是类的每个实例对象独有的变量；

实例变量依附于实例对象，与类本身无关；

实例变量需要实例对象才能访问

**3、什么是继承（4.5.1）**

是一种类与类之间的关系，用于建立类的层次结构；

它体现了 “is - a” 的关系，即子类 “是一种” 父类的特例；

在面向对象程序设计语言中，继承反映了像是世界实体的这种本质联系；

实现了代码的重用。

**4、抽象类和接口的区别（5.4.7）**

抽象类：

可以定义带有方法体的不同方法；一个类只能继承一个抽象父类；抽象类属于一个类的继承体系，并且一般在类体系的顶层；

接口：

接口中的方法都是抽象的（就是方法里面什么都没写）；一个类可以实现多个接口；接口不是类体系的一部分，不相关的类也能实现相同的接口。

# 二、代码题部分

## 1、读代码写结果

public class Example9\_15 {

public static void main(String args[D) {

StringBuffer str=new StringBuffer();

str.append("大家好");

System.out.println("str:"+str);

System.out.println("length:" +str.length();

System.out.println("capacity:"+str.capacity();

StringBuffer sb=new StringBuffer ("Hello");

System.out.println("length:"+sb.length();

System.out.println ("capacity:"+sb.capacity())；

}}

输出：

大家好；length：3；capacity：16；Hello；length：5；capacity：21

有变式题，字符串构造方法里是空的，capacity=16；不是空的就16+leghth

## 2、读代码写注释

public boolean isAbsolute() {} // 判断文件路径是否为绝对路径

public boolean canRead() {} // 检查文件是否可读

public boolean canWrite() {} // 检查文件是否可写

public boolean exists() {} // 判断文件或目录是否存在

public boolean isDirectory() {} // 判断是否是一个目录

public boolean isFile() {} // 判断是否是一个文件

public long length() {} // 获取文件的大小（字节数）

public long lastModified() {} // 获取文件的最后修改时间（时间戳）

public boolean delete() {} // 删除文件或目录，成功返回 true

public String[] list() {} // 获取目录下所有文件和子目录的名称

public boolean mkdir() {} // 创建一个目录，成功返回 true

上述内容，要求知道是什么，考试的时候能写出来意思就行。

**画窗口代码写注释**

import javax.swing.\*; // 导入Swing库，用于创建图形界面

import java.awt.\*; // 导入AWT库，用于颜色和容器操作

public class Example10\_1 {

public static void main(String args[]) {

// 创建一个JFrame窗口，并设置标题为“第一个窗口”

JFrame window1 = new JFrame("第一个窗口");

// 获取窗口的内容面板容器

Container con = window1.getContentPane();

// 设置窗口背景颜色为黄色

con.setBackground(Color.yellow);

// 设置窗口的位置和大小：横坐标60，纵坐标100，宽188，高108

window1.setBounds(60, 100, 188, 108)；

// 设置窗口为可见

window1.setVisible(true)；

// 设置窗口关闭行为为“仅销毁当前窗口”

window1.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}

}

补充：

//创建一个标签对象label，并命名“Hello……”

JLabel label =new JLabel(“Hello Java World”);

con.add(label);//添加label到窗口中

**抽象动物上转型**

abstract class 动物 {// 抽象类“动物”，含抽象方法“cry”，由子类实现具体内容

abstract void cry();

}

class 狗 extends 动物 {// “狗”类继承“动物”类，重写“cry”方法

void cry() {

System.out.println("这是狗的叫声：汪汪．..汪汪");

}

}

class 猫 extends 动物 {// “猫”类继承“动物”类，重写“cry”方法

void cry() {

System.out.println("这是猫的叫声，喵喵．...喵喵...");

}

}

public class Example5\_10 {

public static void main(String args[]) {

动物 animal = new 狗(); // 上转型，用“动物”类型变量指向“狗”类对象

animal.cry();

animal = new 猫(); // 再次上转型，让变量指向“猫”类对象

animal.cry();

}

}

**商标接口**

interface ShowMessage {// 定义接口ShowMessage，有抽象方法“显示商标”

void 显示商标(String s);

}

class TV implements ShowMessage {

// TV类实现ShowMessage接口，实现“显示商标”方法

public void 显示商标(String s) {

System.out.println(s);

}

}

class PC implements ShowMessage {

// PC类实现ShowMessage接口，实现“显示商标”方法

public void 显示商标(String s) {

System.out.println(s);

}

}

public class Example5\_15 {

public static void main(String args[]) {

// 声明ShowMessage接口类型变量sm，用于上转型

ShowMessage sm;

sm = new TV();// 上转型，将TV类对象转为ShowMessage接口类型

sm.显示商标("长城牌电视机");

sm = new PC();// 再上转型，把PC类对象转为接口类型，调用方法体现多态

sm.显示商标("联想奔月5008PC");

}

}

## 3、手写代码

A、分别定义两个字符串S1“我喜欢学习”，S2“数学”，S1,S2串联，并以end结尾赋值给S1再输出

public class Main {

public static void main(String args[]) {

StringBuffer S1 = new StringBuffer("我喜欢学习");

StringBuffer S2 = new StringBuffer("数学");

S1.append(S2);

S1.append("end");

System.out.println(S1);

}

}

B、自定义字符串截取片段

import java.util.Scanner;

public class Demo {

public static void main(String args[]) {

Scanner scan = new Scanner(System.in);

String str = scan.nextLine();

int L = scan.nextInt();

int R = scan.nextInt();

String sub = str.substring(L, R);

System.out.println(sub);

}

}

C、定义一个接口 CanFly， 描述会飞的方法 public void fly();

分别定义类飞机和鸟， 实现 CanFly 接口。

定义一个测试类， 测试飞机和鸟。 测试类中定义一个 makeFly()方法， 让会飞的事物飞起来。

然后在 main 方法中创建飞机对象和鸟对象， 并在 main 方法中调用 makeFly()方法， 让飞机和鸟起飞。

interface Canfly{  
 void fly();  
}  
  
class Bird implements Canfly{  
 public void fly() {  
 System.*out*.println("鸟飞");  
 }  
}  
class Plane implements Canfly {  
 public void fly() {  
 System.*out*.println("飞机飞");  
 }  
}  
  
class TestFly{  
  
 public void testFly(Canfly flyTable)  
 {  
 flyTable.fly();  
 }  
}  
public class Main {  
 public static void main(String[] args)  
 {  
 Bird bird1 = new Bird();  
 Plane plane1 = new Plane();  
 TestFly testFly = new TestFly();  
 testFly.testFly(bird1);  
 testFly.testFly(plane1);  
 }  
}

D、求定义一个 Person 类，包含 name 和 age 成员变量以及输出信息的方法 initInfo()。

定义一个继承自 Person 的 Student 类，增加 studentId 成员变量，并重写 initInfo() 方法以输出包含学号的信息。

在 Test 类的 main 方法中创建一个 Student 对象，姓名为 并调用 initInfo() 方法显示学生的所有信息。

class Person {  
 String name;  
 int age;  
 public Person(String name, int age)  
 {  
 this.name = name;  
 this.age = age;  
 }  
 void initInfo() {  
 System.*out*.println("Name: " + name+", Age: "+age);  
 }  
}  
class Student extends Person {  
 int studentId;  
 public Student(String name, int age, int studentId)  
 {  
 super(name, age);  
 this.studentId = studentId;  
 }  
 void initInfo() {  
 System.*out*.println("Name: " + name+", Age: "+age+", Student ID: "+studentId);  
 }  
}  
  
public class Test {  
 public static void main(String[] args) {  
  
 Student s = new Student("地狱火", 3000, 111111111);  
 s.initInfo();  
 }  
}