1 考试大纲

1.1 考试题型、分值及内容

1.1.1 简答题或程序改错题（2 题*15 分/题=30 分）

① 理解和应用普通类的对象构造过程以及成员方法的设计 (尤其是静态成员和实例成员的设计)。
② 理解和应用构造方法调用规则 。
③ 理解 Object 类的作用。
④ 理解和应用接口。
⑤ 理解和应用简单继承以及构造方法调用链。
⑥ 理解和应用简单异常及其声明、检测、抛出、捕获和处理过程。
⑦ 深刻理解和掌握子类设计过程中重写 Object 类的 toString 和 equals 等方法的作用和目的。
⑧ 理解与应用 this 和 super 关键字。
⑨ 理解面向对象的三大特性，尤其是多态性的应用。
⑩ 理解和掌握内部类/匿名内部类的基础知识和应用原理。
⑪ 熟练掌握基础数据类型和包装数据类型、静态成员和实例成员的概念和应用。

1.1.2 程序阅读题（2 题*15 分/题=30 分）

① 深刻理解和掌握类的定义和对象的构造过程。
② 深刻理解和掌握异常的检测、抛出、捕获和处理过程。
③ 深刻理解和掌握面向对象中多态性概念和应用。
④ 理解 String 类的应用，理解基础类和对应的包装类的应用。
⑤ 理解与应用 Cloneable 接口的作用和应用以及内部机制。
⑥ 深刻理解和掌握常用接口（如 Comparable、Cloneable、Serializable 等）的作用和应用。
⑦ 理解 jdk 中文件 I/O 的类层次结构，掌握常用的文件 I/O 类的使用方法。
⑧ 理解和掌握内部类的基础和使用方法。

1.1.3 程序设计题（3 题，第 1 题 10 分，第 2 题 15 分，第 3 题 15 分，共 40 分）

① 简单类的设计。
② 带继承简单类的设计。

③ 类和接口综合类的设计。
④ 理解 jdk 中常用接口（如 Comparable、Cloneable、Serializable 等）的作用和应用。
⑤ 多态性的应用。

1.2 考试范围和重点

1.2.1 考试范围

教材（《Java 语言程序设计》第 12 版）第 1 章~第 13 章，第 17 章；mooc 中涉及到的相关内容。

1.2.2 考试重点：

对象和类、静态成员和实例成员，继承和多态、异常处理、接口、Object 类、jdk 中的常用接口和类。

2 模拟题集

2.1 简答题或程序改错题

1、指出以下程序的错误，并修改之。

```
public class Test
{
    public static void main(String[] args)
    {
        int[] arr;
        for(int i=0; i<arr.length; i++)
            arr(i) = (int)(Math.random() * 100);
    }
}
```

2、指出以下程序的错误，并修改之。

```
public class Base{
    public static void main(String[] args){
        Derive d = new Derive();
    }
    private int i;
    public Base(int i)    {
        this.i = i;
    }
}
class Derive extends Base{
    private int j;
}
```

3、简述 Java 中 try{}catch{}finally{}的异常运行机制。

4、简述 Object 类中设计 toString 成员方法的目的, 如果子类中没有重写该方法,

Object 类中的默认操作是什么?

5、简述 Java 中多态性的动态绑定机制，并举例说明。

```
public class TestDemo {
    public static void main(String[] args) {//主函数不允许修改
        Person per = new Person("zhangsan", 20);
    }
}
class Person {
    private String name;
    private int age;
    public Person() {
        this.name = "zhangsan";
        this.age = 20;
    }
}
```

6. 简述 Object 类中 public String toString()方法的设计意图，并完成下面的 Date 类的设计。

```
public class TestDemo {
    public static void main(String[] args) {
        Date today = new Date(2018, 5, 26);
        System.out.println(today);//要求此处的输出结果为：2018-5-26
    }
}
class Date {
    private int year;
    private int month;
    private int day;
    public Date(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
}
```

7. 简述以下程序出错的原因，并改正过来。

```
public class TestDemo {
    public static void main(String[] args) {
        Number number = new Number();
    }
}
abstract class Number {
```

```
    public abstract int intValue();
}
class Integer extends Number {
    private int value;
    public int getValue() {
        return value;
    }
    public void setValue(int value) {
        this.value = value;
    }
}
```

8. 观察以下的异常处理伪代码，回答问题。

```
try{
        System.out.println("statement1");
        statement2;
        System.out.println("statement3");
    }
    catch(Exception1 ex1){
        System.out.println("statement4");
    }
    catch(Exception ex2){
        System.out.println("statement5");
    }
    finally{
        System.out.println("statement6");
    }
    System.out.println("statement7");
}
```

（1）如果 statement2 会引起一个 Exception2 的异常，程序的输出结果是什么？

（2）如果 statement2 会引起一个 Exception 的异常，程序的输出结果是什么？

9. 简述 java 中关键字 super 和 this 的作用，并举例说明。

10. 指出以下 java 程序的错误，并改正之。

```
class Test{
    private int id;
    public void Test()    {
        Test(45);
```

```
    }
    public void Test(int id){
        Test.id = id;
    }
}
```

2.2 程序阅读和程序分析题

1、写出以下程序的输出结果。

```
public class Test{
    public static void main(String[] args){
        MyClass t = new MyClass();
        swap(t);
        System.out.println("e1="+t.e1+", e2="+t.e2);
    }
    public static void swap(MyClass t)  {
        int temp = t.e1;
        t.e1 = t.e2;
        t.e2 = temp;
    }
}
class MyClass{
    int e1 = 10;
    int e2 = 20;
}
```

2、写出以下程序的输出结果。

```
public class Test
{
    private static int i = 10;
    private static int j = 20;
    public static void main(String[] args)
    {
        int i = 2;
        int k = 10;
        {
            int j = 6;
            System.out.println("i+j="+i+j);
        }
        k = i + j;
        System.out.println("k="+k);
        System.out.println("j="+j);
```

```
    }
}
```

3、写出以下程序的输出结果。

```
public class Test{
    public static void main(String[] args)    {
        Base b = new Derive();
        System.out.println(b);
        print(b);
    }
    public static void print(Base b){
        System.out.println(b);
    }
}
class Base{
    public Base(){
        System.out.println("Base(int i)");
    }
    @Override
    public String toString(){
        return "Base";
    }
}
class Derive extends Base{
    public Derive(){
        System.out.println("Derive(int i, int j)");
    }
    @Override
    public String toString(){
        return "Derive";
    }
}
```

4、写出以下程序的输出结果。

```
public class Test
{
    public static void main(String[] args)
    {
        String s = "Hello";
        StringBuilder builder = new StringBuilder(s);
        change(s, builder);
        System.out.println(s);
        System.out.println(builder);
    }
```

```
    public static void change(String s, StringBuilder builder)
    {
        s = s + " world";
        builder.append(" world");
    }
}
```

5、写出以下程序的输出结果。

```
public class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("before statement");
            throw new Exception();
            System.out.println("after statement2");
        }
        catch (Exception e)
        {
            System.out.println("catch statement");
        }
        finally
        {
            System.out.println("finally statement");
        }
        System.out.println("end statement");
    }
}
```

6. 阅读以下程序，并写出输出结果。

```
public class TestDemo {
    public static void main(String[] args) {
        Derive derive = new Derive(20);
        Derive derive1 = new Derive();
    }
}
class Base {
    private int i;
    public Base() {
        i = 10;
        System.out.println("Base(): i=" + i);
    }
    public Base(int i) {
```

```
        this.i = i;
        System.out.println("Base(int i): i=" + i);
    }
}
class Derive extends Base {
    private int j;
    public Derive() {
        super();
        j = 10;
        System.out.println("Derive(): j=" + j);
    }
    public Derive(int j) {
        super(20);
        this.j = j;
        System.out.println("Derive(int j): j=" + j);
    }
}
```

7. 阅读以下程序，并写出输出结果。

```
public class TestDemo {
    public static void main(String[] args) {
        Base base = new Base();
        show(base);
        base = new Derive1();
        show(base);
        base = new Derive2();
        show(base);
        Derive2 derive2 = new Derive2();
        show(derive2);
    }
    public static void show(Base base) {
        base.show();
    }
}
class Base {
    public void show() {
        System.out.println("Base::show()");
    }
}
class Derive1 extends Base {
    @Override
    public void show() {
        System.out.println("Derive1::show()");
    }
```

```
}
class Derive2 extends Base {
    @Override
    public void show() {
        System.out.println("Derive2::show()");
    }
}
```

8. 阅读以下程序，并回答问题

```
public class TestDemo {
    public static void main(String[] args) {
        FileInputStream fileInputStream = null;
        try {
            System.out.println("File is openning ...");
            fileInputStream = new FileInputStream(new File("test.dat"));
            System.out.println("File operate succesfully.");
        } catch (Exception e) {
            System.out.println("Open file failed.");
        } finally {
            System.out.println("File is closing ...");
            try {
                if (fileInputStream != null)
                {
                    fileInputStream.close();
                    System.out.println("File is closed.");
                }
            } catch (IOException e) {
                System.out.println("Close file failed.");
            }
        }
    }
}
```

（1）简述 Java 中异常的抛出机制。

（2）如果当前目录下存在文件 test.dat，写出上述程序的输出结果。

9. 在 jdk 中，接口 Cloneable 只是一个标记接口，该接口中没有任何的成员，请简述该接口的作用，并举例说明。

10. 以下程序运行的时候存在运行时错误，会抛出异常，简述抛出异常的原因，

指出并改正过来。

```
public class TestDemo {
    public static void main(String[] args) throws Exception {//主函数内容不允许修改
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream(new File("test.dat")));
        objectOutputStream.writeObject(new A());
        objectOutputStream.close();
    }
}
class A{
}
```

2.3 程序设计题

1、如果两个数组 list1 和 list2 的内容相同，就说它们是相同的。使用下面的方法头编写一个方法，如果 list1 和 list2 相同，输出 true，否则输出 false。

public static boolean equals(int[] list1, int[] list2)

2、通过给文件中的每个字节加 5 来对文件编码。编写一个程序，提示用户输入一个输入文件名和一个输出文件名，然后将输入文件的加密版本存入输出文件。

3、编写一个程序，向一个名为 Exercisel7_05.dat 的文件中存储一个含 5 个 int 值 1、2、3、4、5 的数组，一个表示当前时间的 Date 对象，以及一个 double 值 5.5。

4、创建名为 ComparableCircle 的类，它继承自 Circle 类，并实现 Comparable 接口。实现 compareTo 方法，使其根据面积比较两个圆。编写一个测试程序求出 ComparableCircle 对象的两个实例中的较大者。

5、设计一个名为 Stock 的类。这个类包括：

- 一个名为 symbol 的字符串数据域表示股票代码。
- 一个名为 name 的字符串数据域表示股票名字。
- 一个名为 previousClosingPrice 的 double 型数据域，它存储的是前一日的股票值。

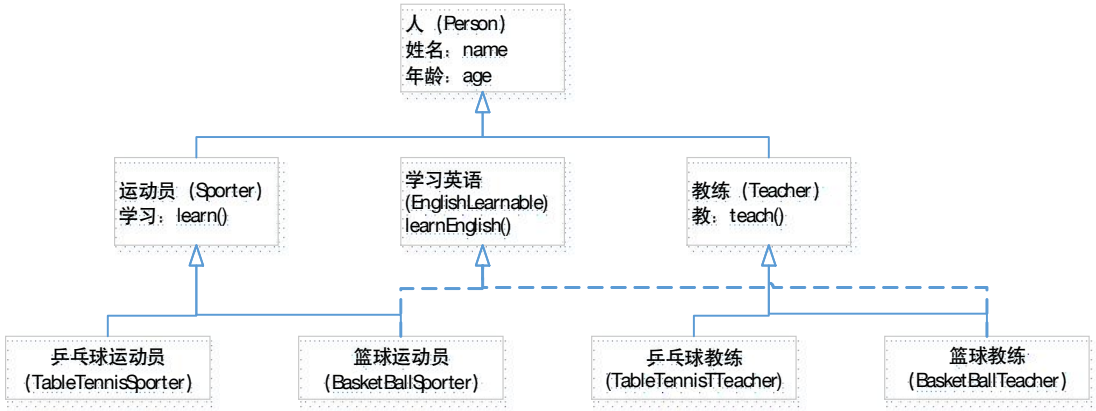
- 一个名为 currentPrice 的 double 型数据域，它存储的是当时的股票值。
- 创建一支有特定代码和名字的股票的构造方法。
- 一个名为 getChangePercent（）的方法，返回从 previousClosingPrice 变化到 currentPrice 的百分比。

实现这个类，并编写一个测试程序，创建一个 Stock 对象，它的股票代码是 ORCL，股票名字为 OracleCorporation，前一日收盘价是 34.5。设置新的当前值为 34.35，然后显示市值变化的百分比。

6、设计一个名为 MyPoint 的类，表示一个带 x 坐标和 y 坐标的点。该类包括：

- 两个带 get 方法的数据域 x 和 y 分别表示它们的坐标。
- 一个创建点(0,0)的无参构造方法。
- 一个创建特定坐标点的构造方法。
- 一个名为 distance 的方法，返回从该点到 MyPoint 类型的指定点之间的距离。
- 一个名为 distance 的方法，返回从该点到指定 x 和 y 坐标的指定点之间的距离。

7、假设有一体育俱乐部，俱乐部中有乒乓球运动员和篮球运动员，以及对应的乒乓球教练和篮球教练。现篮球人员需要出国参加比赛，所以篮球相关人员都需要学习英语。根据以上描述以及下图完成相关类和接口的设计（注意类、抽象类、接口的应用）。



8、假定一个文本文件中包含未指定个数的分数，用空格分开。编写一个 java 程序，提示用户输入文件名称，然后从文件中读入分数，并且显示它们的和以及平均值。

9、根据以下给定的代码完成 Person 类的设计。

```
public class TestDemo {
    public static void main(String[] args) throws Exception {
        List<Person> list = new ArrayList<>();
        for (int i = 0; i < 10; i++) {//向链表中放入 10 个 Person 类型的对象数据
            Person per = new Person("zhangsan" + (new Random()).nextInt(10), 20 + (new Random()).nextInt(10));
            list.add(per);
        }
```

```
        Collections.sort(list);//对链表中的数据按照年龄进行排序

        for (int i = 0; i < 10; i++) {//输出排序后的结果
            System.out.println(list.get(i));
        }
    }
}
```

10、单例模式是最常用到的设计模式之一，其定义就是，单例类的对象只允许一个实例存在, 即一个程序中只要实例化该类, 那么得到的都是这个类同一个对象。

请运用 java，设计一个单例模式的类。