**尚学堂Java 初级软件工程师认证考试试卷**

**答案精讲**

1. **填空题（共20个题目，总计20分）**
2. java.exe
3. 4
4. 2<<3
5. break
6. 方法重载
7. protected
8. this
9. finalize
10. System.exit(1);
11. length
12. 连续
13. int n = Integer.parseInt(“123”);
14. format
15. LinkedList
16. ObjectInputStream
17. System.err
18. Synchronized
19. notify()
20. IP地址
21. 利用反射创建指定类的对象
22. **选择题（共25个题目，总计25分）**
    * 1. AC

分析：Java的跨平台原理决定了其性能比C/C++低；Java语言具有一次编译，到处运行的特点，可以在所有提供并安装了对应虚拟机的平台上运行。

* + 1. D

分析：final是关键字。

* + 1. AB

分析：Java运算符中，赋值运算符级别最低，其次是条件运算符。

* + 1. B

分析：s1=s1+1; 此时需要强制类型转换才可以；但是s1+=1不需要强制转换。

* + 1. B

分析：把子类转换为父类，称为向上转型，自动进行类型转换

* + 1. BD

分析：char可以当做int类型处理

* + 1. B
    2. A

分析：try能单独和finally语句一起使用，而不需要catch

* + 1. A
    2. A

分析：静态代码块在类加载时执行，最早执行，且只执行一次；代码块在每次创建对象时执行，且先于构造方法执行。

* + 1. AD

分析： constInt =constInt+5;错误，因为常量不能再重新赋值；

public int method( );错误，因为和已有方法冲突，违背方法重载规则。

* + 1. B

分析：在change()方法中执行的语句str = "test ok";不会影响main( )中str变量的值，String是不可变字符串；在在change()方法中执行的语句ch[0] = 'g';会影响main()方法中ch的值，因为ch是数组，引用数据类型。

* + 1. AD
    2. D

分析：int result = num1 / num2;出现ArithmeticException异常，不执行try块中后续语句，匹配最后一个catch语句并执行，然后执行finally语句块，最后执行finally块后面的语句。

* + 1. A

分析：可以通过System.out.println()输出三个变量的值，结果为：

sbf=java,C#

sbf1=java,C#

sbf2=java,C#,C#

因为执行StringBuffer sbf1 = sbf.append(",C#");时，sbf和sbf1的值同时变成了java,C#

* + 1. C
    2. AC

分析：一个方法最多也只能有1个可变参数，还必须是最后一个。

可变参数对应的实参可以1个，2个，多个，也能是0个，还可以是一个数组

* + 1. D

分析：float objf=3.4;修改为float objf=3.4f;

i<f.size( )修改为i<f.length

float tmp=(float)list.get(i);修改为float tmp=(Float)list.get(i);

* + 1. C
    2. D

分析：map中key唯一，放入相同key的键值对时，后一个会覆盖前一个。

* + 1. D

分析：fw.close( );需要进行异常处理

* + 1. C
    2. C

分析：应该实现Runnable接口；run方法的修饰符是public。

* + 1. A

分析：HTTP属于应用层协议；TCP和UDP是传输层协议；IP是网络层协议。

* + 1. BD

1. **判断题（共20个题目，总计10分）**
   * 1. √
     2. × Java语言中不同数据类型的长度是固定的，不随机器硬件不同而改变。
     3. √
     4. × 运算符|和&既是逻辑运算符，也是位运算符。
     5. √
     6. × 局部变量在使用前必须进行初始化；成员变量在分配空间时会赋予初始值。
     7. √
     8. × 使用final修饰的方法将无法被子类重写，而不是重载。
     9. × 匿名内部类只能可以实现一个接口，不能实现多个接口。
     10. × ClassCastException也是运行时异常
     11. √
     12. × 数组按照元素值查询某个元素效率低下。
     13. ×  应该是int length=str.length(); String有length方法，而不是length属性
     14. × Collection 接口存储一组不唯一，无序的对象
     15. √
     16. × ByteArrayInutStream和ByteArrayOutputStream是节点流。
     17. √
     18. × Thread.sleep( )方法调用后，当等待时间已到，该线程处于可运行状态。
     19. √
     20. √
2. **简答题（共5个题目，总计25分）**

**1.** **简述static关键字的作用。（5分）**

static可以修饰变量、方法和代码块和内部类，分别称为静态变量，静态方法，静态代码块和静态内部类。

1. 静态变量属于这个类所有，即由该类创建的所有对象共享同一个static变量。
   * 1. 不管有多少个对象，静态变量只有1份。（0.5分）
     2. 静态变量存在方法区中。（0.5分）
     3. 可通过对象名.静态变量名，类名.静态变量名两种方式访问。（0.5分）
     4. 静态变量在第一次使用类的时候分配空间。（0.5分）
2. 静态方法：可通过对象名.静态方法名和类名.静态方法名两种方式访问。（1分）
3. 静态代码块：当类被第一次使用时执行静态代码块，且只被执行一次，主要作用是实现static属性的初始化。（1分）
4. 静态内部类：属于整个外部类，而不是属于外部类的每个对象。（1分）

**2.** **2. 简述接口和抽象类的相同和不同之处。（5分）**

相同点

* 抽象类和接口均包含抽象方法（0.5分）
* 抽象类和接口都不能实例化（0.5分）

两者的区别主要体现在两方面：语法方面和设计理念方面。

语法方面的区别是比较低层次的，非本质的，主要表现在：

* 接口中只能定义全局静态常量。抽象类中可以定义常量和变量（0.5分）
* 接口中所有的方法都是全局抽象方法。抽象类中可以有0个、1个或多个，甚至全部都是抽象方法（0.5分）
* 抽象类中可有构造方法，但不能实例化。接口中不能定义构造方法（0.5分）
* 一个类只能有一个直接父类（可以是抽象类），但可以实现多个接口（0.5分）

二者的主要区别还是在设计理念上，其决定了某些情况下到底使用抽象类还是接口：

* 抽象类体现了一种继承关系，目的是复用代码.父类和子类之间必须存在“is-a”的关系，即父类和子类在概念本质上应该是相同的（1分）
* 接口定义了“做什么”，而实现类负责完成“怎么做”，体现了功能（规范）和实现分离的原则。接口和实现之间可以认为是一种“has-a的关系”（1分）

**3.** **简述List、Set、Collection、Map的区别和联系。（5分）**

1. List和Set是Collection接口的两个子接口（1分）
2. Collection 接口存储一组不唯一，无序的对象（1分）
3. List 接口存储一组不唯一，有序（插入顺序）的对象（1分）
4. Set 接口存储一组唯一，无序的对象（1分）
5. Map接口存储一组键值对象，提供key到value映射。Key无序，唯一。value不要求有序，允许重复。（如果只使用key存储，那就是Set）（1分）

**4. 简述进程和线程的联系和区别。（5分）**

进程和线程的联系：（2分）

1. 线程是指进程内的一个执行单元，也是进程内的可调度实体（0.5分）
2. 一个线程只能属于一个进程，一个进程可有多个线程，至少一个线程（1分）
3. 进程和线程都可以并发执行（0.5分）

线程与进程的区别：（3分）

1. 根本区别：线程是调度和执行基本单位，进程时分配资源的基本单位。（1分）
2. 开销：进程间切换开销大，线程间切换开销小（1分）
3. 所处环境：在操作系统中能同时运行多个任务(进程) 在同一应用程序中有多个顺序流（线程）同时执行（1分）

**5. 简述基于TCP的Socket编程的主要步骤（5分）**

服务器端编程步骤：

* 1. 构建一个ServerSocket实例，指定本地的端口。（1分）
  2. 调用accept()方法来获得客户端的连接请求。通过accept()方法返回的socket实例，建立了一个和客户端的新连接。（1分）
  3. 通过这个返回的socket实例获取InputStream和OutputStream,可以通过这两个stream来分别读和写数据。（0.5分）
  4. 结束的时候调用socket实例的close( )方法关闭socket连接。（0.5分）

客户端编程步骤：

* 1. 构建Socket实例，通过指定的远程服务器地址和端口来建立连接。（1分）
  2. 通过Socket实例包含的InputStream和OutputStream来进行数据的读写。（0.5分）
  3. 操作结束后调用socket实例的close方法，关闭。（0.5分）

1. **编码题（共3个题目，总计20分）**
2. 编写递归算法程序：一列数的规则如下: 0、1、1、2、3、5、8、13、21、34...... 求数列的第40位数是多少。（5分）

**public** **class** TestRecursion {

**public** **static** **void** main(String[] args) {

//调用fibo方法并输出结果（1分）

**int** n = 40;

TestRecursion test = **new** TestRecursion();

System.*out*.println(test.fibo(n));

}

**public** **long** fibo(**int** n){//定义递归方法（1分）

//递归结束条件1：数列的第一个数是0(1分)

**if**(n==1){

**return** 0;

}

//递归的结束条件2：数列的第二个数是1（1分）

**if**(n==2){

**return** 1;

}

//使用递归技术第n个数（n>2）（1分）

**return** fibo(n-1)+fibo(n-2);

}

}

1. 编写多线程程序，模拟多个人通过一个山洞的模拟。这个山洞每次只能通过一个人，每个人通过山洞的时间为5秒，有10个人同时准备过此山洞，显示每次通过山洞人的姓名和顺序。（6分）

**public** **class** MulThreadTest{

**public** **static** **void** main(String[] args) {

//创建一个山洞(0.5分)

Tunnel tul = **new** Tunnel();

//创建十个过山洞线程（1.0分）

Thread p1 = **new** Thread(tul,"p1");

Thread p2 = **new** Thread(tul,"p2");

Thread p3 = **new** Thread(tul,"p3");

Thread p4 = **new** Thread(tul,"p4");

Thread p5 = **new** Thread(tul,"p5");

Thread p6 = **new** Thread(tul,"p6");

Thread p7 = **new** Thread(tul,"p7");

Thread p8 = **new** Thread(tul,"p8");

Thread p9 = **new** Thread(tul,"p9");

Thread p10 = **new** Thread(tul,"p10");

//启动十个线程0.5分

p1.start(); p2.start();

p3.start(); p4.start();

p5.start(); p6.start();

p7.start(); p8.start();

p9.start(); p10.start();

}

}

**class** Tunnel **implements** Runnable{

**private** **int** crossedNum = 0;//初始人数0.5分

**public** **void** run(){//1分

cross();

}

**public** **synchronized** **void** cross(){

**try** { //每个人通过山洞的时间为5秒（1分）

Thread.*sleep*(5000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

//人数计数（0.5分）

crossedNum++;

//显示一下每次通过山洞人的姓名（1分）

System.*out*.println(Thread.*currentThread*().getName()+

"通过了山洞，这是第"+crossedNum+"个用户");

}

}

评分标准：可以采用其他方法，比如extends Thread实现。只要能够实现功能，都按照步骤给分。每个步骤得分如上所示。

1. 由控制台按照固定格式输入学生信息，包括学号，姓名，年龄信息，当输入的内容为exit退出；将输入的学生信息分别封装到一个Student对象中，再将每个Student对象加入到一个集合中，要求集合中的元素按照年龄大小正序排序；最后遍历集合，将集合中学生信息写入到记事本，每个学生数据占单独一行。（9分）

**public** **class** Student **implements** Comparable<Student>{

**private** Integer num;

**private** String name;

**private** Integer age;

**//省略getter和setter方法**

**//省略构造方法**

**public** **int** compareTo(Student stu) {

**return** **this**.age-stu.age;

}

**public** String toString() {

**return** "Student [age=" + age + ", name=" + name +

", num=" + num + "]";

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Set<Student> stuSet = *saveStudentInfo*();

*outputInfo*(stuSet);

}

**private** **static** Set<Student> saveStudentInfo() {

Scanner input = **new** Scanner(System.*in*);

// 保存学生信息的TreeSet集合对象

Set<Student> stuSet = **new** TreeSet<Student>();

**while** (**true**) {

// 输入提示

System.*out*.println("请输入学生信息:(学号#姓名#年龄)");

String inputData = input.nextLine();

// 判断是否退出 inputData.equals("exit")

**if** ("exit".equals(inputData)) {

**break**;

}

// 将用户输入的学生信息分割为String[]

String[] info = inputData.split("#");

// 将输入信息封装到Student对象中

Student stu =

**new** Student(Integer.*parseInt*(info[0]), info[1],

Integer.*parseInt*(info[2]));

// 将学生对象加入集合

stuSet.add(stu);

}

**return** stuSet;

}

**private** **static** **void** outputInfo(Set<Student> stuSet) {

File file = **new** File("e:/student.txt");

// 创建文件输出流对象

FileWriter fw = **null**;

**try** {

fw = **new** FileWriter(file);

Iterator<Student> it = stuSet.iterator();

**while** (it.hasNext()) {

String info = it.next().toString();

// 将info字符串，写入记事本

fw.write(info);

// 完成换行功能

fw.write("\r\n");

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**try** {

fw.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

评分标准：

正确编写学生类（2分）， 正确完成学生信息输入并放入集合（4分），正确将集合内容写入文件（3分）。具体得分点如下：

1. 创建Student类，提供三个属性和setter/getter方法，提供构造方法（1分）
2. Student类实现Comparable，重写compareTo( )，按照年龄正序排列（1分）
3. 通过控制台输入多个Student信息。格式规定为：编号#姓名#年龄（0.5分）
4. 当输入的内容为exit退出。（0.5分）
5. 将用户输入的学生信息使用split( )分割为String[]。（1分）
6. 将输入信息封装到Student对象中。（1分）
7. 创建TreeSet对象，并将每个Student对象放入TreeSet。（1分）
8. 创建和关闭输出流对象。（1分）
9. 正确的遍历TreeSet集合。（1分）
10. 将每个学生信息写入文件。（1）

此题目的难点主要有：

实现Comparable接口指定按照年龄正序排列；放入TreeSet会按照规则自动排序

使用split( )分割字符串；创建输出流写学生信息到文件并换行。

如果没有以上思路实现，只要能完成功能即可，按照相应功能步骤给分。