****

**尚学堂JavaSE 软件工程师**

**认证考试试卷**

**笔试**

考试时间150分钟

总分 100分

姓 名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

身份证号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

准考证号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

考试结束后考卷必须收回，否则考试成绩无效

**尚学堂JavaSE软件工程师认证考试试卷**

1. **填空题（共20个题目，总计20分）**
2. 使用Java开发应用程序包括编写源程序，编译源程序，解释并运行三个步骤，其中编译和解释分别需要使用javac.exe和 java.exe 实现。
3. Java中整型变量有byte、short、int和long四种，不同类型的整数变量在内存中分配的字节数不同，数值范围也不同。对于int型变量，内存分配 4 个字节。
4. 使用位运算符来实现运算效率最高，所以最有效率的方法算出2乘以8等于多少的语句是 2<<3 。
5. 在任何循环语句的主体部分， break 关键字用于强行退出当前循环，忽略循环体中任何其它语句和循环条件测试，会执行当前循环下边的语句。
6. 重载 是指在一个类中可以有多个方法具有相同的名字，但这些方法的参数类型、个数、顺序必须不同，与方法的返回值和访问修饰符无关。
7. 使用权限修饰符 protect 修饰的类的成员变量和成员方法，可以被当前包中所有类访问，也可以被它的子类（同一个包以及不同包中的子类）访问。
8. 使用关键字 this 来调用同类的其它构造方法，优点同样是以最大限度地代码的利用程度 ，减少程序的维护工作量。
9. 垃圾回收机制回收任何对象之前，总会先调用它的 finalized() 方法，该方法是Object类提供的方法。不要主动调用某个对象的该方法，应该交给垃圾回收机制调用。
10. 异常处理中finally块可确保无论是否发生异常，该块中代码总能被执行。finally块不执行的唯一情况是在异常处理代码中执行 System.exit() 语句退出Java虚拟机。
11. 要获取一个数组的长度，可以通过 length 属性来获取，但获取的只是为数组分配的空间的数量，而不是数组中实际已经存放的元素的个数。
12. 数组会在内存中开辟一块 固定长度 的空间，每个空间相当于之前的一个变量，称为数组的元素。数组的长度一经确定，就无法再改变。
13. 包装类Integer的静态方法可以将字符串类型的数字”123”转换成基本整型变量n，其实现语句是： int 3 = Integer.parseInt(“123”); 。
14. DateFormat类可以实现日期和指定格式字符串之间相互转换，其中将日期转换成指定格式字符串的方法是\_\_\_\_\_format()\_\_\_\_\_\_\_。
15. \_\_\_\_\_\_\_\_\_LinkedList\_\_\_\_是一种集合类，它 采用链表作为的存储结构，便于删除和添加元素，但是按照索引查询元素效率低下。
16. Java IO体系中，\_\_ObjectInputStream\_\_\_\_\_是字节输入流，不仅提供了存取所有Java基础类型数据（如：int，double 等）和String的方法,也提供了提供存取对象的方法。
17. Java系统的标准输入对象是System.in，标准输出对象有两个，分别是标准输出System.out和标准错误输出\_\_\_\_\_\_\_System.err\_\_\_\_\_\_\_\_。
18. 多线程访问某个共享资源可能出现线程安全问题，此时可以使用\_\_synchronized\_关键字来实现线程同步，从而避免安全问题出现，但会影响性能，甚至出现死锁。
19. 在线程通信中，调用wait( )可以是当前线程处于等待状态，而为了唤醒一个等待的线程，需要调用的方法是\_\_\_\_\_\_\_notify()\_\_\_\_\_\_\_。
20. \_\_\_\_\_\_Ip地址\_\_\_\_用来标志网络中的一个通信实体的地址。通信实体可以是计算机，路由器等。
21. Class.forName('com.bjsxt.stumgr.entity.Student').newInstance( ); 语句的作用是\_\_\_\_ \_\_\_\_\_new Student()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
22. **选择题（共25个题目，总计25分）**

|  |  |  |
| --- | --- | --- |
| **1)** | **以下选项中关于Java跨平台原理的说法正确的是（ a c ）。（选择二项）** | |
|  |  |  |
|  | **A** | Java源程序要先编译成与平台无关的字节码文件(.class)，然后字节码文件再被解释成机器码运行 |
|  | **B.** | Java的跨平台原理决定了其性能比C/C++高 |
|  | **C.** | Java虚拟机是可运行Java字节码文件的虚拟计算机。不同平台的虚拟机是不同的，但它们都提供了相同的接口 |
|  | **D.** | Java语言具有一次编译，到处运行的特点，可以在所有的平台上运行 |

|  |  |  |
| --- | --- | --- |
| **2)** | **在Java中，以下错误的变量名是（ d ）。（选择一项）** | |
|  |  |  |
|  | **A** | constant |
|  | **B.** | flag |
|  | **C.** | a\_b |
|  | **D.** | final |

|  |  |  |
| --- | --- | --- |
| **3)** | **以下Java运算符中优先级别最低的两个选项是（ a c ）。（选择二项）** | |
|  |  |  |
|  | **A** | 赋值运算符= |
|  | **B.** | 条件运算符 ?= |
|  | **C.** | 逻辑运算符| |
|  | **D.** | 算术运算符+ |

|  |  |  |
| --- | --- | --- |
| **4)** | **关于以下Java程序中错误行的说明正确的是（ a ）。（选择一项）** | |
|  | **public** **class** Test2 {  **public** **static** **void** main(String[] args) {  **short** s1=1; //1  s1=s1+1; //2  s1+=1; //3  System.*out*.println(s1);  }  } | |
|  |  |  |
|  | **A** | 1行错误 |
|  | **B.** | 2行错误 |
|  | **C.** | 3行错误 |
|  | **D.** | 1行，2行，3行都错误 |

|  |  |  |
| --- | --- | --- |
| **5)** | **关于Java中的多态，以下说法不正确的为（b ）。（选择一项）** | |
|  |  |  |
|  | **A** | 多态不仅可以减少代码量，还可以提高代码的可扩展性和可维护性 |
|  | **B.** | 把子类转换为父类，称为向下转型，自动进行类型转换 |
|  | **C.** | 多态是指同一个实现接口，使用不同的实例而执行不同的操作 |
|  | **D.** | 继承是多态的基础，没有继承就没有多态 |

|  |  |  |
| --- | --- | --- |
| **6)** | **下列选项中关于变量x的定义，（c d ）可使以下switch语句编译通过。（选择二项）** | |
|  | **switch**(x) {  **case** 100 :  System.*out*.println("One hundred");  **break**;  **case** 200 :  System.*out*.println("Two hundred");  **break**;  **case** 300 :  System.*out*.println( "Three hundred");  **break**;  **default** :  System.*out*.println( "default");  } | |
|  |  |  |
|  | **A** | double x = 100; |
|  | **B.** | char x = 100; |
|  | **C.** | String x = "100"; |
|  | **D.** | int x = 100; |

|  |  |  |
| --- | --- | --- |
| **7)** | **以下Java程序编译运行后的输出结果是（ B ）。（选择一项）** | |
|  | **public** **class** Test {  **public** **static** **void** main(String[] args) {  **int** i = 0, sum = 0;  **while** (i <= 10) {  sum += i;  i++;  **if** (i % 2 == 0)  **continue**;  }  System.*out*.println(sum);  }  } | |
|  |  |  |
|  | **A** | 0 |
|  | **B.** | 55 |
|  | **C.** | 50 |
|  | **D.** | 36 |

|  |  |  |
| --- | --- | --- |
| **8)** | **以下选项中能单独和finally语句一起使用的是（ a ）。（选择一项）** | |
|  |  |  |
|  | **A** | try |
|  | **B.** | catch |
|  | **C.** | throws |
|  | **D.** | throw |

|  |  |  |
| --- | --- | --- |
| **9)** | **给定如下Java代码，编译运行的结果是（ ）。（选择一项）** | |
|  | **public** **class** Test {  **public** **static** **void** main(String[] args) {  **int** sum=0;  **for**(**int** i=1;i<10;i++){  **do**{  i++;  **if**(i%2!=0)  sum+=i; 8  }**while**(i<6);  }  System.*out*.println(sum);  }  } | |
|  |  |  |
|  | **A** | 8 |
|  | **B.** | 15 |
|  | **C.** | 24 |
|  | **D.** | 什么也不输出 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **10)** | **以下Java程序编译运行后的输出结果是（ a ）。（选择一项）** | | | |
|  | **class** HelloA {  **public** HelloA( ) {  System.*out*.println("HelloA");  }  { System.*out*.println("I'm A class");  }  **static** {  System.*out*.println("static A");  }  }  **public** **class** HelloB **extends** HelloA {  **public** HelloB( ) {  System.*out*.println("HelloB");  }  { System.*out*.println("I'm B class");  }  **static** {  System.*out*.println("static B");  }  **public** **static** **void** main(String[] args) {  **new** HelloB( );  }  } | | | |
|  |  |  | | |
|  | **A** | static A  static B  I'm A class  HelloA  I'm B class  HelloB | C. | static A  I'm A class  HelloA  static B  I'm B class  HelloB |
|  | **B.** | static A  static B  I'm A class  I'm B class  HelloA  HelloB | D | static A  static B  HelloA  HelloB  I'm A class  I'm B class |

|  |  |  |
| --- | --- | --- |
| **11)** | **以下选项可替换题目中//add code here而不产生编译错误的是（a d ）。（选择二项）** | |
|  | **public** **abstract** **class** MyClass{  **public** **int** constInt=5;  //add code here  **public** **void** method( ){  }  } | |
|  |  |  |
|  | **A** | **public** **abstract** **void** method(**int** a); |
|  | **B.** | constInt =constInt+5; |
|  | **C.** | **public** **int** method( ); |
|  | **D.** | **public** **abstract** **void** anotherMethod( ); |

|  |  |  |
| --- | --- | --- |
| **12)** | **分析下面的Java程序，编译运行后的输出结果是（ d ）。（选择一项）** | |
|  | **public** **class** Example {  String str = **new** String("good");  **char**[] ch = { 'a', 'b', 'c' };  **public** **static** **void** main(String args[]) {  Example ex = **new** Example( );  ex.change(ex.str, ex.ch);  System.*out*.print(ex.str + "and");  System.*out*.print(ex.ch);  }  **public** **void** change(String str, **char** ch[]) {  str = "test ok";  ch[0] = 'g';  }  } | |
|  |  |  |
|  | **A** | goodandabc |
|  | **B.** | goodandgbc |
|  | **C.** | test okandabc |
|  | **D.** | test okandgbc |

|  |  |  |
| --- | --- | --- |
| **13)** | **在Java中，以下程序段能正确为数组赋值的是（ a c ）。（选择二项）** | |
|  |  |  |
|  | **A** | int a[]={1,2,3,4}; |
|  | **B.** | int b[4]={1,2,3,4}; |
|  | **C.** | int c[];c={1,2,3,4}; |
|  | **D.** | int d[];d=new int[]{1,2,3,4}; |

|  |  |  |
| --- | --- | --- |
| **14)** | **编译并运行如下Java程序，将输出（ d ）。（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {  **try** {  **int** num1 = 2;  **int** num2 = 0;  **int** result = num1 / num2;  System.*out*.println(result);  **throw** **new** NumberFormatException( );  } **catch** (ArrayIndexOutOfBoundsException e) {  System.*out*.print("1");  } **catch** (NumberFormatException e) {  System.*out*.print("2");  } **catch** (Exception e) {  System.*out*.print("3");  } **finally** {  System.*out*.print("4");  }  System.*out*.print("5");  } | |
|  |  |  |
|  | **A** | 134 |
|  | **B.** | 2345 |
|  | **C.** | 1345 |
|  | **D.** | 345 |

|  |  |  |
| --- | --- | --- |
| **15)** | **给定如下Java代码，编译运行的结果是（ a ）。（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {  StringBuffer sbf = **new** StringBuffer("java");  StringBuffer sbf1 = sbf.append(",C#");  String sbf2 = sbf + ",C#";  System.*out*.print(sbf.equals(sbf1));  System.*out*.println(sbf2.equals(sbf));  } | |
|  |  |  |
|  | **A** | true false |
|  | **B.** | true true |
|  | **C.** | false false |
|  | **D.** | false true |

|  |  |  |
| --- | --- | --- |
| **16)** | **给定如下Java代码，编译运行的结果是（ c ）。（选择一项）** | |
|  | **public** **static** **void** main(String []args){  String s1= **new** String("pb\_java\_OOP\_T5");  String s2 = s1.substring(s1.lastIndexOf("\_"));  System.*out*.println("s2="+s2);  } | |
|  |  |  |
|  | **A** | s2=\_java\_OOP\_T5 |
|  | **B.** | s2=\_OOP\_T5 |
|  | **C.** | s2=\_T5 |
|  | **D.** | 编译出错 |

|  |  |  |
| --- | --- | --- |
| **17)** | **以下选项中关于Java中方法的可变参数的说法正确的是（a c ）。（选择二项）** | |
|  |  |  |
|  | **A** | 可变参数是JDK1.5增加的内容，用来表示方法的形参 |
|  | **B.** | 一个方法可以没有可变参数，可以有1个或者多个可变参数 |
|  | **C.** | 可变参数可以被当作数组参数来处理 |
|  | **D.** | 可变参数对应的实参可以1个，2个，多个，但不能0个，还可以是一个数组 |

|  |  |  |
| --- | --- | --- |
| **18)** | **以下Java代码的作用是首先将一个数组的内容存入集合，然后判断集合中是否有指定的元素存在，其中共有（ c ）处错误。（选择一项）** | |
|  | **import java.util.List;**  **public** **class** Test {  **public** **int** getIndexofArray(**float**[] f){  **int** rtn=-1;  **float** objf=3.4; \*  List list=**null**;  **for**(**int** i=0;i<f.size( );i++){  list.add(f[i]); \*  }  **for**(**int** i=0;i<list.size( );i++){  **float** tmp=(**float**)list.get(i);  **if**(objf==tmp){  rtn=i;  }  }  **return** rtn;  }  } | |
|  |  |  |
|  | **A** | 0 |
|  | **B.** | 1 |
|  | **C.** | 2 |
|  | **D.** | 3 |

|  |  |  |
| --- | --- | --- |
| **19)** | **在Java中，以下File类的方法中（ c ）用来判断是否是目录。（选择一项）** | |
|  |  |  |
|  | **A** | isFile( ) |
|  | **B.** | getFile( ) |
|  | **C.** | isDirectory( ) |
|  | **D.** | getPath( ) |

|  |  |  |
| --- | --- | --- |
| **20)** | **给定如下Java代码，编译运行的结果是（ c ）。（选择一项）** | |
|  | **public** **class** Test{  **public** **static** **void** main(String[] args){  Map<String,String> map=**new** HashMap<String,String>( );  String s="code";  map.put(s,"1");  map.put(s,"2");  System.*out*.println(map.size( ));  }  } | |
|  |  |  |
|  | **A** | 编译时发生错误 |
|  | **B.** | 运行时引发异常 |
|  | **C.** | 正确运行，输出：1 |
|  | **D.** | 正确运行，输出：2 |

|  |  |  |
| --- | --- | --- |
| **21)** | **阅读下列Java IO代码，在有标注的四行代码中，有编译错误的是第（ d ）处。（选择一项）** | |
|  | **import** java.io.\*;  **public** **class** Test{  **public** **static** **void** main(String[] args){  String str = "Hello World";  FileWriter fw = **null**;  **try**{  fw = **new** FileWriter("c:\\hello.txt"); //1  fw.write(str); //2  }**catch**(IOException e){  e.printStackTrace( ); //3  }**finally**{  fw.close( ); //4  }  }  } | |
|  |  |  |
|  | **A** | 1 |
|  | **B.** | 2 |
|  | **C.** | 3 |
|  | **D.** | 4 |

|  |  |  |
| --- | --- | --- |
| **22)** | **当线程调用start( )后，其所处状态为（ c ）。（选择一项）** | |
|  |  |  |
|  | **A** | 阻塞状态 |
|  | **B.** | 运行状态 |
|  | **C.** | 就绪状态 |
|  | **D.** | 新建状态 |

|  |  |  |
| --- | --- | --- |
| **23)** | **如下代码创建一个新线程并启动线程，问:四个选项中可以保证正确代码创建target对象，并能编译正确的是（ c ）？（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {  Runnable target=**new** MyRunnable( );  Thread myThread=**new** Thread(target);  } | |
|  |  |  |
|  | **A** | **public** **class** MyRunnable **extends** Runnable {  **public** **void** run( ) {  }  } |
|  | **B.** | **public** **class** MyRunnable **extends** Runnable {  **void** run( ) {  }  } |
|  | **C.** | **public** **class** MyRunnable **implements** Runnable {  **public** **void** run( ) {  }  } |
|  | **D.** | **public** **class** MyRunnable **implements** Runnable {  **void** run( ) {  }  } |

|  |  |  |
| --- | --- | --- |
| **24)** | **以下协议中属于TCP/IP协议栈中应用层协议的是（ a ）。（选择一项）** | |
|  |  |  |
|  | **A** | HTTP |
|  | **B.** | TCP |
|  | **C.** | UDP |
|  | **D.** | IP |

|  |  |  |
| --- | --- | --- |
| **25)** | **Java UDP Socket编程主要用到的两个类是（ b d ）。（选择二项）** | |
|  |  |  |
|  | **A** | UDPSocket |
|  | **B.** | DatagramSocket |
|  | **C.** | UDPPacket |
|  | **D.** | DatagramPacket |

1. **判断题（共20个题目，总计10分，正确√，错误×）**
2. 注释是程序员为读者作的说明，是提高程序可读性的一种手段。Java注释不会出现在字节码文件中，即Java编译器编译时会跳过注释语句。（ **√** ）
3. Java语言中不同数据类型的长度是不固定的，随机器硬件不同而改变。（ ）
4. char类型变量可以存储一个Unicode字符，在内存中占2个字节，该字符可以是一个汉字。（ **√** ）
5. 运算符||和&&既是逻辑运算符，也是位运算符。根据两侧操作数的类型判断是逻辑运算符还是位运算符。（ **√** ）
6. do-while循环结构的特点是先循环再判断，循环体至少执行一次。（ **√** ）
7. Java中所有的变量，不管是成员变量还是局部变量，在使用前都必须进行初始化。（ ）
8. 如果定义一个类时没有使用extends，则它的父类是java.lang.Object。（ **√** ）
9. 使用final修饰的变量将变成常量，其中不能再被改变；使用final修饰的方法将无法被子类重载；使用final修饰的类将无法再被其他类继承。（ **√** ）
10. 匿名内部类适合创建那种只需要使用一次的类，它可以实现一个或多个接口，或者继承一个父类。（ ）
11. NullPointerException是运行时异常，ClassCastException是Checked异常。（ ）
12. 声明数组并分配空间后，数组的每个元素将会赋予初始值。（ **√**）
13. 数组的主要优点是按照索引查找某个元素效率高，同时按照元素值查询某个元素效率也很高，但是添加和删除元素需要大量移动元素，效率低下。（ ）
14. 执行语句String str="abcedf"; int length=str.length; 后，能够得到字符串的长度是6。（ ）
15. Collection 接口存储一组不唯一，有序的对象，它有两个子接口：List和Set。（ **√** ）
16. HashSet类底层采用哈希表存储结构，特点是查询速度快，但是其中的元素无序排列。（ **√** ）
17. ByteArrayInutStream和ByteArrayOutputStream对内存中的字节数组进行读写操作，属于字节流，属于处理流而不是节点流。 （ ）
18. 实现Serializable接口的可以被序列化和反序列化。该接口中没有定义抽象方法，也没有定义常量。（ **√** ）
19. Thread.sleep( )方法调用后，当等待时间未到，该线程所处状态为阻塞状态。当等待时间已到，该线程所处状态为运行状态。（ ）
20. TCP协议一种面向连接的、可靠的、基于字节流的通信协议 。HTTP、FTP、TELNET、SMTP 都是基于TCP协议的应用层协议。（ **√** ）
21. 在Java中，只要给定类的名字，那么就可以通过 Java反射技术来获得类的所有信息，进而创建对象和执行方法。（ **√** ）
22. **简答题（共5个题目，总计25分）**
23. 简述static关键字的作用。

提示：从static可以修饰变量，方法，代码块，内部类四个方面来回答。（5分）

1. 简述接口和抽象类的相同和不同之处。（5分）
2. 简述List、Set、Collection、Map的区别和联系。（5分）
3. 简述进程和线程的联系和区别。（5分）
4. 简述基于TCP的Socket编程的主要步骤。

提示：分别说明服务器端和客户端的编程步骤。（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）
3. 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）
4. 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）

推荐步骤：

1. 创建Student类，并指定按照年龄正序排列
2. 通过控制台输入多个不同Student信息。格式规定为：编号#姓名#年龄
3. 取出字符串中相应信息放入Student对象，并将Student加入到集合中
4. 遍历集合的过程中将学生的信息输入到记事本

难点：

1. 如何指定学生按照年龄正序排列
2. 如果从字符串“编号#姓名#年龄”中提取学生信息
3. 放入哪种集合后可以保证学生按照年龄大小正序排列
4. 如何将集合中学生信息写入记事本，每个学生数据占单独一行