**尚学堂Java 初级软件工程师认证考试试卷精解**

1. **填空题（共20个题目，总计20分）**
2. path
3. 文档注释
4. 2
5. &&
6. continue
7. int n=(int)3.14159;
8. 堆内存
9. 对象名.静态变量名
10. package import class
11. 运行时异常(或RuntimeException)
12. length
13. format
14. enum
15. Comparable
16. next()
17. DataInputStream
18. 序列化
19. invoke
20. 阻塞
21. 端口（号）
22. **选择题（共25个题目，总计25分）**
    1. A

分析：编译时使用javac Test.java，源文件的扩展名.java不可少；解释时使用java Test，运行的是类而非class文件，所以类名后不能加扩展名.class。

* 1. CD

分析：Java标识符命名规则

标识符必须以字母、\_下划线、美元符$开头；

标识符其它部分可以是字母、下划线“\_”、美元符“$”和数字的任意组合；

Java 标识符大小写敏感，且长度无限制；

不可以是Java的关键字；

public是关键字，3num以数字开头。

* 1. B

分析：m = false 是赋值语句，而不是m == false，所以m的值是false。

* 1. A
  2. D

分析：do-while循环先循环后判断，循环体至少被执行一次；while循环先判断后循环，循环体可能一次也不执行；while和do-while循环执行效率没有差别。

* 1. D

分析：任何两种循环都可以相互嵌套；可以任意层次循环，但是一般不超过3层；外层循环变量变化一次，内层循环变量要变化一遍。

* 1. B

分析：main方法中基本数据类型变量i作方法的实在参数，在mb\_operate中会重新分配新的空间存放形参i，不会影响到main方法中i的值。

* 1. A

分析：构造方法的第一条语句如果不是显式的通过this调用当前类的构造方法，或显式的通过super调用父类的构造方法，则默认是super()，可以省略，作用是调用父类无参数的构造方法；所以会先向上追溯到Object，然后再依次向下执行各级父类的构造方法，直到当前类为止。

* 1. B

分析：一个Java类没有定义构造器，则系统会自动定义一个无参的构造方法。如果已定义，则系统不会添加无参数构造方法，所以Java类至少有一个构造方法。

* 1. BD

分析：静态static方法中不能出现this和super关键字，因为this和super只有在创建对象后才会在堆内存中分配空间，而可以在没有创建对象之前就是要static变量和static方法

* 1. C

分析：static方法只能出现static变量，不能出现非static变量；非static方法中可以出现static变量和非static变量。

* 1. C

分析：使用final修饰变量，变量的值将无法修改，从而变为常量。此题中使用final修饰的是person，而不是name，所以B正确，C出现编译错误。

* 1. C

分析： finally块的具体执行过程：执行try或catch中代码；遇到return/throw后先执行finally中语句块；执行return/throw。

finally块中语句不执行的唯一情况：执行System.exit(1)退出Java虚拟机

* 1. B

分析：

==：

a) 基本类型，比较的是值

b) 引用类型，比较的是地址

c) 不能比较没有父子关系的两个对象

equals()

a) 系统类一般已经覆盖了equals()，比较的是内容。

b) 用户自定义类如果没有覆盖equals()，将调用父类的equals（比如是Object），而Object的equals的比较是地址（return (this == obj);）

c) 用户自定义类需要覆盖父类的equals()

注意：Object的==和equals比较的都是地址，作用相同

* 1. D

分析： StringBuffer是字符串“变量”，其值可以直接修改，String是字符串“常量”，其值不能直接修改，只能通过修改引用来指向新的字符串来实现修改。

x.append(y);修改的是x的内容，y = x；修改的是y的引用。

该题目难度指数高，涉及String、StringBuffer，引用类型参数的方法调用，锻炼价值很大，可以借助内存分配图来理解。

* 1. AD

分析：定义二维数组并分配空间时，=左侧不能指定数组长度，=右侧可以同时指定两个维度的长度，也可以只指定第一个维度的长度。

* 1. BD

分析：int的默认值是0，Integer是引用数据类型，默认值是null；Integer i=5;该语句在JDK1.5之后可以正确执行，使用了自动装箱箱功能，而不是自动拆箱功能。

* 1. AC

分析：注意Collections和Collection的区别，List接口和Set接口是Collection接口有两个子接口。

Collection中存放的元素具有无序，不唯一的特点，List中存放的元素具有有序，不唯一的特点，而HashSet中存放的元素具有无序，唯一的特点。

* 1. B

分析：HashSet中存放的元素具有无序，唯一的特点，所以长度是2，而不是4。

* 1. C

分析：Vector和ArrayList实现原理相同，功能相同，都是长度可变的数组结构，很多情况下可以互用。

两者的主要区别如下： Vector是早期集合类，ArrayList是替代Vector的；Vector线程安全；ArrayList重速度轻安全，线程非安全，性能较高。

HashMap和Hashtable实现原理相同，功能相同，底层都是哈希表结构，查询速度快，在很多情况下可以互用。

两者的主要区别如下：Hashtable是早期JDK提供，HashMap是新版JDK提供；Hashtable线程安全，性能较低，HashMap线程非安全，性能较高。

* 1. AD

分析：FileReader和FileWriter是节点流，直接从文件读写数据；BufferedReader和BufferedWriter是处理流，提供缓冲区功能，提高读写效率。

* 1. C

分析：InputStreamReader和OutputStreamWriter是转换流，采用了适配器设计模式，可以将字节流转换成字符流。而在各种节点流和包装流之间则采用了装饰模式，装饰模式是继承的一种有效替代方案，避免产生大量的子类。

* 1. A

分析：定义线程有两种方式，继承Thread类和实现Runnable接口。创建线程对象的方式也有所不同。因为Runnable接口的实现类本身并不是线程，所以需要作为Thread类的参数来传递。此题目采用第二种方式来定义线程和创建线程对象。

* 1. A

分析：在多个线程访问同一个资源时，可以使用synchronized关键字来实现线程同步，保证对资源安全访问，但是也会导致性能下降，甚至导致死锁。

* 1. AD

分析：TCP协议一种面向连接（连接导向）的、可靠的、基于字节流的运输层通信协议 。特点：面向连接；点到点的通信；高可靠性；占用系统资源多、效率低。

UDP是一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。特点:

非面向连接，传输不可靠，可能丢失；发送不管对方是否准备好，接收方收到也不确认；可以广播发送；非常简单的协议，开销小。

1. **判断题（共20个题目，总计10分）**
2. √
3. ×Java的字节码文件的扩展名是class，是一种平台独立的二进制文件。
4. √
5. √
6. √
7. × 在多重循环中，外层循环变量变化一次，内层循环变量要变化一遍。
8. × 程序调用自身的编程技巧称为递归，其特点是编程简单但是占用内存空间多。
9. √
10. √
11. × 将子类对象赋给父类引用变量，称为向上转型，将无法访问子类特有的方法。
12. √
13. √
14. √
15. ×数组中的元素可以是任何数据类型，包括基本类型和引用类型。集合中的元素只能是引用类型，基本类型数据需要转换为对应包装类的对象才能放入集合。
16. √
17. √
18. × Collections类是专门用来操作集合的工具类，提供一系列静态方法实现对各种集合的操作。
19. × 一个File对象可以代表一个文件或目录，它可以获取文件和目录属性，不可以访问文件内容。
20. × 线程Thread是进程内部的一个执行单元，它是程序中一个单一顺序控制流程。
21. √
22. **简答题（共5个题目，总计25分）**
    1. **方法重载和方法重写（覆盖）的区别。（4分）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 英文 | 位置不同 | 修饰符 | 返回值 | 方法名 | 参数 | 抛出异常 | 方法体 |
| 重载 | overload | 同一个类中 | 无关 | 无关 | 相同 | 不同 | 无关 | 不同 |
| 重写 | override | 子类和父类间 | 大于等于 | 小于等于 | 相同 | 相同 | 小于等于 | 不同 |

* 1. **写出java.lang.Object类的六个常用方法的声明并说明其作用。（6分）**

1. public boolean **equals**(java.lang.Object) 比较内容
2. public native int hashCode() 哈希码
3. public java.lang.String toString() 变成字符串
4. public final native java.lang.Class getClass() 获取类结构信息
5. protected void finalize() throws java.lang.Throwable 垃圾回收前执行的方法
6. protected native Object clone() throws java.lang.CloneNotSupportedException 克隆
7. public final void wait() throws java.lang.InterruptedException 多线程中等待功能
8. public final native void notify() 多线程中唤醒功能
9. public final native void notifyAll() 多线程中唤醒所有等待线程的功能
   1. **接口和抽象类的联系和区别。（5分）**

相同点

* 抽象类和接口均包含抽象方法，类必须实现所有的抽象方法，否则是抽象类
* 抽象类和接口都不能实例化，他们位于继承树的顶端，用来被其他类继承和实现

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，不能定义变量。抽象类中可以定义常量和变量。
* 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。
* 抽象类中可以有构造方法，但不能用来实例化，而在子类实例化是执行，完成属于抽象类的初始化操作。接口中不能定义构造方法。
* 一个类只能有一个直接父类（可以是抽象类），但可以充实实现多个接口。一个类使用extends来继承抽象类，使用implements来实现接口。

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，抽象类中定义了各个子类的相同代码，可以认为父类是一个实现了部分功能的“中间产品”，而子类是“最终产品”。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。
* 接口并不要求实现类和接口在概念本质上一致的，仅仅是实现了接口定义的约定或者能力而已。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. **Vector和ArrayList、ArrayList和LinkedList的区别和联系。（6分）**

**Vector和ArrayList的区别和联系**

实现原理相同，功能相同，都是长度可变的数组结构，很多情况下可以互用

两者的主要区别如下

1. Vector是早期JDK接口，ArrayList是替代Vector的新接口
2. Vector线程安全，ArrayList重速度轻安全，线程非安全
3. 长度需增长时，Vector默认增长一倍，ArrayList增长50%

**ArrayList和LinkedList的区别和联系**

ArrayList实现了长度可变的数组，在内存中分配连续空间。遍历元素和随机访问元素的效率比较高；



LinkedList采用链表存储方式。插入、删除元素时效率比较高



* 1. 面向对象设计原则有哪些。（4分）

面向对象设计原则是面向对象设计的基石，面向对象设计质量的依据和保障，设计模式是面向对象设计原则的经典应用

1. 单一职责原则SRP
2. 开闭原则OCP
3. 里氏替代原则LSP
4. 依赖注入原则DIP
5. 接口分离原则ISP
6. 迪米特原则LOD
7. 组合/聚合复用原则CARP

开闭原则具有理想主义的色彩，它是面向对象设计的终极目标。其他设计原则都可以看作是开闭原则的实现手段或方法。

1. **编码题（共3个题目，总计20分）**
   1. 写出饿汉式单例模式、懒汉式单例模式、多线程情况下懒汉式双重判断单例模式的示例代码。（7分）

/\*\*

\* 饿汉式的单例模式，在类加载时创建单例实例，而不是等到第一次请求实例时创建

\* 1、私有 的无参数构造方法Singleton()，避免外部创建实例

\* 2、私有静态属性instance \* 3、公有静态方法getInstance()

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance* = **new** Singleton();

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**return** *instance*;

}

}

/\*\*

\* 懒汉式的单例模式

\*在类加载的时候不创建单例实例，只有在第一次请求实例的时候的时候创建

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance*;

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**if**(*instance*==**null**){ //只创建一次对象

*instance* = **new** Singleton();

}

**return** *instance*;

}

}

/\*\*

\*多线程情况下懒汉式的单例模式，避免创建多个对象

\* 需要使用**synchronized关键字和**进行两次null判断

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance*;

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**if**(*instance* ==**null**){//避免每次加锁，只有第一次没有创建对象时才加锁

**synchronized**(Singleton.**class**){//加锁，只允许一个线程进入

**if**(*instance*==**null**){ //只创建一次对象

*instance* = **new** Singleton();

}

}

}

**return** *instance*;

}

}

* 1. 写出使用JDBC访问数据库表t\_student中所有数据，并将结果集内容封装成List返回的代码。（6分）

**public** List<Student> findAll() {

String driver="oracle.jdbc.driver.OracleDriver";

String url="jdbc:oracle:thin:@localhost:1521:orcl";

String username="bjsxt";

String password="bjsxt";

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

List<Student> stuList = **new** ArrayList<Student>();

**try** {

//加载驱动

Class.*forName*(driver);

//获取连接

conn = DriverManager

.*getConnection*(url, username,password);

//建立会话

stmt = conn.createStatement();

//发送SQL命令得到结果

rs = stmt.executeQuery("select \* from t\_student");

//处理结果

**while**(rs.next()){

//取出结果集中各列数据

**int** id = rs.getInt("id");

String name = rs.getString("name");

**int** age = rs.getInt("age");

**double** score = rs.getDouble("score");

Date enterDate = rs.getDate("enterDate");

//封装成Student对象

Student stu=**new** Student(id name age, score,enterDate);

//加入集合

stuList.add(stu);

}

} **catch** (SQLException e) {

e.printStackTrace();

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

}**finally**{

//关闭资源(依次关闭rs，stmt，conn)

super.closeAll(rs,stmt,conn);

}

**return** stuList;

}

* 1. 使用TCP网络编程完成用户登录功能。（7分）

**public** **class** LoginServer {

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

// 1.创建一个ServerSocket，指定等待端口

ServerSocket serverSocket = **new** ServerSocket(8888);

// 2.使用ServerSocket接收用户的请求（处于监听状态）

Socket socket = serverSocket.accept();

// 3.创建输入流和输出流

ObjectInputStream ois =

**new** ObjectInputStream(socket.getInputStream());

DataOutputStream dos =

**new** DataOutputStream(socket.getOutputStream());

// 4.接收用户数据并输出

User user = (User) ois.readObject();

System.*out*.println("我是服务器，收到的信息是：" + user);

// 5.发送反馈信息

**boolean** flag = **false**;

**if** ("bjsxt".equals(user.getUsername())

&& "bjsxt".equals(user.getPassword())) {

flag = **true**;

}

dos.writeBoolean(flag);

// 6.关闭流和socket

ois.close();

dos.close();

socket.close();

}

}

**public** **class** LoginClient {

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

// 1.创建Socket，用来发送请求

Socket socket =

**new** Socket(InetAddress.*getByName*("127.0.0.1"), 8888);

// 2.创建输入流和输出流

DataInputStream dis =

**new** DataInputStream(socket.getInputStream());

ObjectOutputStream oos =

**new** ObjectOutputStream(socket.getOutputStream());

// 3.发送数据

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

System.*out*.println("请输入用户名");

String username = input.nextLine();

System.*out*.println("请输入密码");

String password = input.nextLine();

User user = **new** User(username, password);

oos.writeObject(user);

// socket.shutdownOutput();

// 5.接收反馈信息并输出

**boolean** flag = dis.readBoolean();

**if** (flag) {

System.*out*.println("登录成功");

} **else** {

System.*out*.println("登录失败");

}

// 4.关闭流和socket

dis.close();

oos.close();

socket.close();

}

}