**填空题**

1、Java是 面向对象 语言，类是可观事务的 抽象 而对象是类的实例。

2、Java语言的主要特点有 简单性、安全性、面向对象、分布式、可移植、多线程等。

3、面向对象程序设计语言的三大基本特征是是：封装、继承、多态。

4、所有的 Java程序由**public static void main(String[] args)**方法开始执行。

5、Java枚举是一个特殊的类，一般表示一组常量Java枚举类使用 enum关键字来定义，各个常量使用 逗号 ， 来分割。

6、Java 源程序文件和字节码文件的扩展名分别是.java和.class 。

7、一般java程序的类体由两部分组成：一部分是属性，另一部分是方法 。

8、Java中成员变量用final修饰符修饰的是常量。

9、在类中，this指\_当前对象\_\_\_，super指**\_**父类对象**\_\_\_**。

10、用static修饰的变量叫静态变量，它可以通过\_类的对象\_和\_类名\_两种方式来访问。

11、子类对父类继承来的属性重新定义称为\_\_重写\_\_。子类对自身拥有的同名方法的重新定义称\_重载\_\_\_。

12、所有的类都是Object的子类，Java的窗口通常是JFrame\_\_的子类，异常类都是\_Exception\_的子类，抽象类只能定义但不能被实例化。

13、Java语言的基本数据类型有（整数）int、short、int、long、char（字符）、（浮点数）float、double、（布尔型）boolean(true,false)

14、float 和 double 都是表示浮点型的数据类型，它们之间的区别在于精确度的不同。

15、Java语言中提供了两种类型的字符串类来处理字符串，它们是String\_和StringBuffer类。

16、continue 语句是跳过循环体中剩余的语句而强制执行下一次循环，其作用为结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判定。

17、 类 是java程序中的基本结构单位

18、把一个子类对象的引用向上转型为父类当使用向上转型时，可以使用父类引用调用子类重写过的方法，这个过程称为**多态**。

19、多态存在的三个必要条件继承、重写、父类引用指向子类对象。

20、在子类的构造方法中可以通过super关键字调用父类的构造方法。

用关键字abstract修饰的类称为抽象类。

21、如果一个类前面加上**final**修饰符，这个类就不能被其他类继承。

如果一个实例方法前面加上**final**修饰符，该方法就不能被重写。

1. Java语言使用接口(interface)来模拟和实现多继承。

23、使用关键字**interface**定义接口，使用关键字implements实现接口。

24、接口中的方法默认会自动加上**abstract**关键字。

25、一个类只能继承一个父类，但却能实现多个接口！

26、如果需要使用的类和当前类在**同一个包**中，可以省略包的名称。

如果需要使用的类和当前类**不在同一个包中**，就需要使用关键字

**import**导入该类。

27、每一个.java文件里面 只能有一个类 声明为public。

28、一个类可以包含多个构造方法，但必须保证参数各不相同。

29、Web的mvc三层结构中的m,v,c分别代表：model,view,controller*****。*****

30、MyBatis执行SQL的两种方式

第一种方式：通过**SqlSession**提供的方法

第二种方式：**使用DAO接口动态代理**

31、一般上，从服务器获取数据使用get方式，向服务器发送数据使用post方式，****post方式传输数据相对比较安全****

32、JSON (JavaScript Object Notation)，是一种基于文本、独立于编程语言的轻量级的数据交换格式。

33、Java解析json常用的第三方框架有： Gson 、Jackson 、Fastjson

34、Java中的代理分为动态代理和静态代理 。

35、java.io包提供大量和输入输出相关的流类。

36、所有输入流都是抽象类InputStream或抽象类Reader的子类。InputStream用于字节输入 Reader用于字符输入 **。**

所有输出流都是抽象类OutputStream或抽象类Writer的子类。

OutputStream用于字节输出 ，Writer用于字符输出

37、使用输入流（输出流同理）通常包括4个基本步骤：

(1) 设定输入流的源 (2) 创建指向源的输入流

(3) 让输入流读取源中的数据 (4)关闭输入流。

38、使用instanceof判断一个对象是否是某个类或其子类所创建的对象。

39、通过异常处理，我们可以对在程序中的错误进行控制和提示，以防程序崩溃。

40、使用break  语句可以直接强行退出当前的循环，忽略循环体中任何其他语句和循环条件测试。

**简答题：**

**一、使用“==”与使用equals方法来进行比较有什么区别？**

**答：**使用equals方法来比较对象在意义上的相等。使用“==”比较两个引用变量，是判别两个引用变量是否指向相同的对象。

**二、使用文件字节输入流和文件字节输出流将源目录中的一个文件拷贝到目标文件夹中：**

**1、如果文件尺寸过大，不能一次性拷贝，如何解决？**

**2、能否提供拷贝进度提示？**

**答：**分批拷贝，可以用while循环的方式将文件内容读到一个byte字节数组中（长度可自定义），再将这个数组中的数据写入目标文件直到文件内容全部被读取完。显示进度可以先获取文件的大小length，再计算字节数组占文件大小的百分比，每次循环对已读取的数据长度累加，打印输出已读取的长度占文件总大小的百分比即可。

**三、简述public、protect、private的区别?**

答：1、public表明该数据成员、成员函数是对所有用户开放的，所有用户都可以直接进行调用。2、private表示私有，除了类自己之外，都不可以访问使用。3、protected对于自身、子类、同一个包中的类可以访问，没有限制，而对于没有继承关系的其他的外部则不能访问。

**四、抽象类和接口的区别：**

相同点：

1、都不能被实例化。

2、实现接口的或继承抽象类的普通类都必须重写所有的抽象方法。

区别：

1.抽象类中可以有一些非抽象方法的存在，但是接口要求只能包含抽象方法，接口不能存在方法的实现。

2.实现抽象类使用extends关键字来继承抽象类，使用关键implements来实现接口。

3.抽象类是属于类，有构造方法，接口是所有抽象方法的集合，不是类，没有构造方法。

4.一个类可以只能继承一个抽象类，但可以实现多个接口。

5.抽象类中可以包含静态方法，接口中不能。

6.抽象类中的抽象方法除了private都可以，而接口中的默认public abstract。

7.抽象类中的属性可以是任意类型的修饰符，接口中只能是public static final。

8.接口用interface定义，强调特定功能的实现，而抽象类用abstract class定义，强调所属关系。

**五、什么是泛型，为什么要使用泛型?**

**答**：泛型的本质是把参数的类型参数化，也就是所操作的数据类型被指定为一个参数，这种参数类型可以用在类、接口和方法中。使用泛型编写的程序代码，要比使用 Object 变量再进行强制类型转换的代码，具有更好的安全性和可读性。多种数据类型执行相同的代码使用泛型可以复用代码。

**六、简述使用MyBatis的基本流程**

**答：**第一步：根据数据库表的结构定义对应的实体类。

第二步：根据对数据库表的操作定义对应的DAO接口。

第三步：根据DAO接口定义对应的实体-关系映射xml文件，在xml文件中定义与DAO接口对应的SQL语句。

第四步：对DAO接口中的每一个方法做单元测试。

**七、为什么需要控制反转与依赖注入？**

**答**：如果一个类里面需要用到其他的类，我们称这个类对其他类存在“依赖”，或者 这个类和其他类是“耦合”的。如果设计的软件系统是“强耦合”的，那么这个软件系统的“可伸缩性”和“可扩展性”就会受到限制。在设计软件系统时我们希望能够尽可能提高系统的“可伸缩性”和“可扩展性。

### **八、封装的优点**

**答**：在面向对象程式设计方法中，封装是指一种将抽象性函式接口的实现细节部分包装、隐藏起来的方法。有如下优点：

1. 良好的封装能够减少耦合。

2. 类内部的结构可以自由修改。

3. 可以对成员变量进行更精确的控制。

4. 隐藏信息，实现细节。

**九、注解有什么用？**

**答**：1生成文档。这是最常见的。

2跟踪代码依赖性，实现替代配置文件功能。

1. 在编译时进行格式检查。如@Override

**十、简述JDK、JRE、JVM之间的关系**

**答**：JDK包含了Java的运行环境（即JRE）和Java工具。JRE包含了一个Java虚拟机（JVM）以及一些标准的类别函数库。总的来说，JDK、JRE、JVM三者都处在一个包含关系内，JDK包含JRE，而JRE又包含JVM。

具体地讲：

JDK = JRE + 开发工具集（例如Javac编译工具等）

JRE = JVM + Java SE标准类库

**补充知识点**

**1、什么是node.js？**

node.js基于 Chrome V8 引擎的 JavaScript 运行环境

底层网络通信使用libuv，是一个基于事件驱动、非阻塞式 I/O 的模型,使得JavaScript能够运行在服务器端。使用npm进行包的管理.

**2、什么是开-闭原则**

答：所谓“开-闭原则”指的是让设计的系统应当对扩展开放，对修改关闭。在设计系统时，应当首先考虑到用户需求的变化，将应对用户变化的部分设计为对扩展开放，而设计的核心部分是经过精心考虑之后确定下来的基本结构，这部分应当是对修改关闭的，即不能因为用户的需求变化而再发生变化，因为这部分不是用来应对需求变化的。如果系统的设计遵守了“开-闭原则”，那么这个系统一定是易维护的，因为在系统中增加新的模块时，不必去修改系统中的核心模块。

**3、什么是Mybatis？**

**MyBatis**是一款优秀的**持久层框架**，它支持自定义SQL、存储过程以及高级映射。 **MyBatis**对JDBC代码以及设置参数和获取结果集的工作进行了封装，使开发者只需要关注SQL语句本身，而无需关注注册驱动，创建连接等繁杂过程。**MyBatis**可以通过简单的**XML或注解**来配置和映射原始类型、接口和 Java POJO（Plain Old Java Objects）为数据库中的记录。

**4、异常处理**：

异常处理是指对程序中可能出现错误的代码进行异常捕获等一系列操作。

异常处理分三类

第一类，特殊异常，即自己业务上的特殊异常，需要特别处理的，直接在catch中将该异常处理吃掉。比如有个业务需要访问一个鉴权的接口，当第一次鉴权不成功时，捕获到异常，在catch上继续鉴权。

第二类，可忽略的异常。即有些异常抛出不会影响正常流程和业务的，可忽略掉，在catch里不作处理

第三类，需要交给上一层处理的异常。比如在service层抛出的异常，需要再往上抛出去给controller层，让其处理或者发给前端。

## **5、进程与线程**

## 进程是指一个内存中运行的应用程序，每个进程都有自己独立的一块内存空间，即进程空间或（虚空间）。进程不依赖于线程而独立存在，一个进程中可以启动多个线程。比如在Windows系统中，一个运行的exe就是一个进程。线程是指进程中的一个执行流程，一个进程中可以运行多个线程。比如java.exe进程中可以运行很[多线程](https://so.csdn.net/so/search?from=pc_blog_highlight&q=%E5%A4%9A%E7%BA%BF%E7%A8%8B" \t "https://blog.csdn.net/yjsz2010/article/details/_blank)。线程总是属于某个进程，线程没有自己的虚拟地址空间，与进程内的其他线程一起共享分配给该进程的所有资源。

简而言之：

一个程序至少有一个进程，一个进程至少有一个线程。

线程的划分尺度小于进程，使得多进程程序的并发性高。

另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

**6、JDK、JRE、JVM分别是什么？**

JDK，全称Java Development Kit，是 Java 语言的软件开发工具包，主要用于移动设备、嵌入式设备上的Java应用程序。JDK是整个Java开发的核心。

JRE，全称Java Runtime Environment，是指Java的运行环境，是可以在其上运行、测试和传输应用程序的Java平台。

JVM，全称Java Virtual Machine（Java虚拟机），是一种用于计算设备的规范，它是一个虚构出来的计算机，引入JVM后，Java语言在不同平台上运行时不需要重新编译。JVM是Java跨平台的核心。 **7、Spring框架**

Spring框架是[Java平台](https://baike.baidu.com/item/Java%E5%B9%B3%E5%8F%B0/3793459" \t "https://baike.baidu.com/item/Spring%20Boot/_blank)上的一种开源应用框架，提供具有控制反转特性的容器。Spring框架具有[面向切面编程](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%BC%96%E7%A8%8B/6016335" \t "https://baike.baidu.com/item/Spring%20Boot/_blank)（AOP）框架，有流行的数据访问框架中提供支持，同时还可以与Spring的事务管理一起使用，为数据访问提供了灵活的抽象。SpringBoot基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

**SpringBoot所具备的特征有**：

（1）可以创建独立的[Spring](https://baike.baidu.com/item/Spring/85061" \t "https://baike.baidu.com/item/Spring%20Boot/_blank)应用程序，并且基于其Maven或Gradle插件，可以创建可执行的JARs和WARs；（2）内嵌Tomcat或Jetty等Servlet容器；（3）提供自动配置的“starter”项目对象模型（POMS）以简化[Maven](https://baike.baidu.com/item/Maven/6094909" \t "https://baike.baidu.com/item/Spring%20Boot/_blank)配置；（4）尽可能自动配置Spring容器；

（5）提供准备好的特性，如指标、健康检查和外部化配置；（6）绝对没有代码生成，不需要XML配置。

**8、JDBC**

Java数据库连接，（Java Database Connectivity，简称JDBC）是[Java语言](https://baike.baidu.com/item/Java%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/Java%E6%95%B0%E6%8D%AE%E5%BA%93%E8%BF%9E%E6%8E%A5/_blank)中用来规范客户端程序如何来访问数据库的[应用程序接口](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F%E6%8E%A5%E5%8F%A3/10418844" \t "https://baike.baidu.com/item/Java%E6%95%B0%E6%8D%AE%E5%BA%93%E8%BF%9E%E6%8E%A5/_blank)，提供了诸如查询和更新数据库中数据的方法。我们通常说的JDBC是面向关系型数据库的。

**9、JavaFX**

JavaFX是Java平台中使用XML+CSS构建图形用户界面的技术。

JavaFX是一个开源的下一代客户端应用程序平台，用于构建在Java上的桌面、移动和嵌入式系统。

**附录**

## Java 关键字

|  |  |  |
| --- | --- | --- |
| **类别** | **关键字** | **说明** |
| 访问控制 | private | 私有的 |
| protected | 受保护的 |
| public | 公共的 |
| default | 默认 |
| 类、方法和变量修饰符 | abstract | 声明抽象 |
| class | 类 |
| extends | 扩充,继承 |
| final | 最终值,不可改变的 |
| implements | 实现（接口） |
| interface | 接口 |
| native | 本地，原生方法（非 Java 实现） |
| new | 新,创建 |
| static | 静态 |
| strictfp | 严格,精准 |
| synchronized | 线程,同步 |
| transient | 短暂 |
| volatile | 易失 |
| 程序控制语句 | break | 跳出循环 |
| case | 定义一个值以供 switch 选择 |
| continue | 继续 |
| default | 默认 |
| do | 运行 |
| else | 否则 |
| for | 循环 |
| if | 如果 |
| instanceof | 实例 |
| return | 返回 |
| switch | 根据值选择执行 |
| while | 循环 |
| 错误处理 | assert | 断言表达式是否为真 |
| catch | 捕捉异常 |
| finally | 有没有异常都执行 |
| throw | 抛出一个异常对象 |
| throws | 声明一个异常可能被抛出 |
| try | 捕获异常 |
| 包相关 | import | 引入 |
| package | 包 |
| 基本类型 | boolean | 布尔型 |
| byte | 字节型 |
| char | 字符型 |
| double | 双精度浮点 |
| float | 单精度浮点 |
| int | 整型 |
| long | 长整型 |
| short | 短整型 |
| 变量引用 | super | 父类,超类 |
| this | 本类 |
| void | 无返回值 |

## 算术运算符

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **例子** |
| + | 加法 - 相加运算符两侧的值 | A + B 等于 30 |
| - | 减法 - 左操作数减去右操作数 | A – B 等于 -10 |
| \* | 乘法 - 相乘操作符两侧的值 | A \* B等于200 |
| / | 除法 - 左操作数除以右操作数 | B / A等于2 |
| ％ | 取余 - 左操作数除以右操作数的余数 | B%A等于0 |
| ++ | 自增: 操作数的值增加1 | B++ 或 ++B 等于 21（区别详见下文） |
| -- | 自减: 操作数的值减少1 | B-- 或 --B 等于 19（区别详见下文） |

## 关系运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| == | 检查如果两个操作数的值是否相等，如果相等则条件为真。 | （A == B）为假。 |
| != | 检查如果两个操作数的值是否相等，如果值不相等则条件为真。 | (A != B) 为真。 |
| > | 检查左操作数的值是否大于右操作数的值，如果是那么条件为真。 | （A> B）为假。 |
| < | 检查左操作数的值是否小于右操作数的值，如果是那么条件为真。 | （A <B）为真。 |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真。 | （A> = B）为假。 |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真。 | （A <= B）为真。 |