Java期末复习题练习汇总

1.什么是静态变量/成员变量/局部变量？有何性质？

**静态变量：使用static关键字在类级别声明的变量。**

性质：存储在类加载的内存区域，与类的任何对象实例无关；生命周期与类相同，从类首次被加载到内存开始，到程序结束被卸载；可通过类名直接访问，也可通过对象访问，一般使用前者，因为后者不属于对象的状态；可用于保存跨对象的信息和存储对象个数；

**成员变量：在类中但在方法外部声明的变量，没有static关键字。**

性质：每个对象实例都有自己的成员变量副本，存储在堆内存中；生命周期与对象实例相同，对象被创建时初始化，对象不再被使用并被垃圾收集时销毁；必须通过对象实例访问；用于描述对象的状态或属性；

**局部变量：在方法内部或者任何代码块（如for循环，if条件等）中声明的变量。**

存储在栈内存中，生命周期短，从声明的代码块执行时开始，当代码块执行结束后就结束，局部变量会从栈上移除；只能在声明它的代码块中访问；可用于临时计算或控制程序流。

2.什么是overload和override？有什么区别？

**Overload是方法重载**：当同一个类中存在两个或更多的方法具有相同的名称，但参数列表不同时发生方法的重载，允许方法根据不同的参数执行不同的功能。要求参数列表必须不同，可通过改变参数的数量，类型及顺序实现，对返回类型及访问修饰符没有特定要求；

**Override是方法重写**：当子类定义一个与父类具有相同名称和参数列表的方法时，该子类方法会覆盖父类方法，称为方法重写。实现运行时多态，子类可以根据需要提供特定的实现，返回类型可以是相同的或是父类方法返回类型的子类型，不能拥有比父类更严格的访问修饰符。

**区别**：方法重载是编译时多态的一种形式，而方法重写是运行时多态的一种形式；重载发生在同一个类中或者接口中，方法名称相同，但参数不同；重写基于子类继承，子类中的方法与父类中的方法具有相同名称和参数列表的方法覆盖父类的方法；重写的目的是修改或扩展类的行为，而重载的目的是增加更多的方法类处理不同类型的输入。

3.public，private，protected，default的访问权限

Protected比较特殊，功能如下：

文本

描述已自动生成

表格

描述已自动生成

4. 在异常处理的过程中，try，catch 和 finally 各有什么作用？

try块用来包裹可能产生异常的代码，如果在try块内的代码抛出了一个异常，那么异常可被catch块来捕获。try块之后至少要有一个catch块或一个finally块。

catch块用来捕获和处理try块内抛出的异常。可以有一个或多个catch块跟在try块后面，捕获不同类型的异常。

finally块包含无论异常是否发生，是否被捕获，否会执行的代码。

5. 什么是继承/封装/多态？

继承：一种创建新类的方式，新类继承了现有类（父类）的属性和方法。这种机制可以实现代码复用，允许新创建的子类扩展和修改父类的行为。

封装：将对象的数据（属性）和操作这些数据的代码（行为）捆绑在一起的过程，可以看作是对对象内部的隐藏和保护。这意味着内部工作方式可以从外部代码中隐藏起来，只暴露出一个清晰的接口供外部操作。

多态：指允许不同类的对象对同一消息做出响应的特性。在编程中，多态意味着同一个操作可能会因为操作对象的不同而表现出不同的行为。

6. 请解释 this/super 关键字的使用方法。（chat）

This关键字用于指代当前对象，即调用方法或构造器的那个对象实例，可用于以下场景：

（1）引用当前类的实例变量

文本

描述已自动生成

（2）调用当前类的其他构造器

文本

描述已自动生成

（3）传递当前的实例

文本

描述已自动生成

（4）返回当前的实例

文本

中度可信度描述已自动生成

Super关键字用于指代当前对象的父类版本，用于访问父类的成员（变量和方法），可用于以下场景：

（1）引用父类的变量

文本

描述已自动生成

（2）调用父类的方法

文本

描述已自动生成

（3）调用父类的构造器

文本

描述已自动生成

7. 说明抽象类和接口有什么区别。

（1）抽象类通常作为多个类的共同基类，实现一些通用的方法和属性；接口定义了一组方法规范，强制实现类遵循某些行为；

（2）抽象类可以有成员变量，包括非final和非static的；接口中的变量都默认为是public static final的常量；

（3）抽象类可以有完全实现的方法（具体方法），也可以有抽象方法；接口在Java7之前只能包含抽象方法，自Java8以后，接口可以包含静态方法和默认方法，自Java9之后还可以包含私有方法；

（4）抽象类可以有构造方法，接口不能有构造方法；

（5）子类使用extends来继承抽象类，并可以通过一个抽象类来实现多个类的通用行为；类使用implements来实现接口，一个类可以实现多个接口；

（6）抽象类中的方法可以有public、protected和private这样的访问级别；接口中的所有方法的访问级别都是public（默认）的，Java9以后可以含有私有方法；

（7）Java不支持通过类来实现多继承，因此一个类只能继承一个抽象类；接口支持多继承，一个类可以同时实现多个接口；

（8）抽象类是为了代码复用和扩展的途径，通常用来表示“是一个”的关系；接口通常表示“能做什么”或“像是一个”的能力。

1. 列举 Java 中常用的字节输入流和字节输出流并说明其特点，至少 3 对。
2. FileInputStream和FileOutputStream：

FileInputStream用于从文件中读取数据；FileOutputStream用于向文件中写入数据；它们是处理文件数据的基本类。FileOutputStream在写入数据时会覆盖文件内容，除非追加指定模式。

1. BufferedInputStream和BufferedOutputStream：

BufferedInputStream在读取数据时提供了一个内部缓冲区；BufferedOutputStream在写出数据时也提供了一个内部缓冲区；

这对提供缓冲区的读写操作，可以提高I/O操作的效率。因为减少了与磁盘的直接交互次数。

1. DataInputStream和DataOutputStream：

DataInputStream允许你以便携方式读取基本类型数据（如int，float，double等）；DataOutputStream允许你以便携方式写入基本数据类型数据；

这对流适合读写数据而不仅是字节，它们支持基本的数据类型操作。

1. 多线程中 start 方法和 run 方法分别有什么作用？

（1）start（）方法：

Start（）方法是Thread类的一个方法，当你调用一个线程的start()方法时，Java虚拟机会创建一个新的线程，这各线程会执行run（）方法；

调用start（）方法后不会立即执行run（）方法里的代码，而是使得该线程变为可运行状态（Runnable）之后该线程由JVM调度，并最终执行线程的run（）方法；

Strart（）方法不能被多次调用。一但线程启动之后，再次调用start（）方法会抛出异常。

1. run（）方法

Run（）方法是一个需要被重写的方法，它定义了线程的操作和功能。它是在新线程调用的，是线程被启动后用来执行特定任务的代码；如果直接调用线程中的run（）方法，他不会创建新的线程，而是在当前线程中同步运行run（）方法中的代码，也就是说，如果直接调用run（）方法，它将作为一个普通方法被调用，和多线程无关。

1. 简述 Java 中的异常处理机制的简单原理和应用。

异常的处理机制是一种强大的错误处理方式，它可以在程序运行时识别和响应异常情况。Java异常处理的基本原理是将异常情况封装在对象中，并在出现问题的程序部分抛出这些对象，然后在调用堆栈中寻找合适的处理器（catch块）来处理这些对象。

异常处理的应用主要涉及以下4个关键字：

1. try：这是编程时尝试执行的代码块，其中可能包含引发异常的代码。
2. Catch：紧跟在try块后面的是一个或多个catch块，用于捕获并处理try块中出现的异常。Catch块指定了要捕获的异常类型和当异常发生时如何处理它们。
3. Finally：这是一个可选的代码块，紧跟在最后一个catch块之后，并且无论发生异常与否，其总是执行。
4. Throw：用于在代码中的任何位置手动抛出异常现象。
5. Throws：在方法签名中，用于声明一个方法可能会抛出一个或多个异常，这告诉调用者需要对这些异常进行处理。

11.简述 throw 和 throws 关键字的区别。

Throw：throw是一个语句，用于在代码的任意位置手动抛出一个异常实例；通常与异常类的新实例一起使用，如throw new SomeExpection（）；。Throw抛出的异常需要被捕获处理（使用try/catch），或在当前方法的申明中用throws关键字向上抛出。Throw可以抛出已检查的（checked）异常或未检查（unchecked）异常。



Throws出现在方法签名中，用于声明一个方法在执行过程中可能会抛出的一个或多个已检查的异常。这表示调用该方法的代码需要处理这些异常，不管是通过try/catch块捕获异常处理，还是在调用方法上继续声明throws。当方法中有可能抛出已检查异常，但并未在该方法内部处理时，必须在方法签名中使用throws。Throws后面跟的是异常类型，如果由多个异常可能会被抛出，则用逗号分离。

12.简述静态方法与非静态方法的区别？

（1）声明和调用：静态方法是使用static关键字声明的，属于类本身，而不是类的特定实例；非静态方法没有static关键字，属于类的实例。

（2）调用方式：静态方法可以通过类名直接调用，不需要创建类的实例；非静态方法必须通过类的实例进行调用。

（3）访问权限：静态方法可以访问类的其他静态成员，但不能直接访问非静态成员。

（4）使用上下文：静态方法通常用于工具或帮助方法，以及那些不依赖对象状态的操作；非静态方法通常与特定实例的状态相关联，它们可以访问和修改实例的成员变量。

（5）覆盖差异：静态方法不能被覆盖（override），在继承时它们是被隐藏的；非静态方法可以被子类覆盖，形成多态。

13. final 关键字的作用。

（1）变量：当final修饰变量时，该变量成为一个常量，一旦给它赋初值后，就不能再改变它的值了。

（2）方法：当final修饰方法时，该方法不能被子类覆盖，保证了方法的不变性。

（3）类：当final修饰类时，表明这个类不能被继承。

14.实现多线程的方法。

（1）继承Thread类：通过继承Thread类并覆盖其run方法来创建一个新的线程类。然后创建实例并调用start方法来启动线程。



（2）实现Runnable接口：需要实现run方法，并将Runnable对象传递给Thread对象。然后通过调用Thread对象的start方法启动线程。



使用Runnable接口的优势是它允许类继承其他类，并可用于实现lambda表达式，因为Runnable是一个函数式接口。

15.什么是构造方法？构造方法有哪些特点？

构造方法是一种特殊的方法，用来初始化对象。当通过关键字new创建一个新的对象时，构造方法被调用，并为对象分配内存空间，同时设置对象的初始状态。

特点：名称和类名相同；没有返回类型，自动调用，可有参数，不能被static，final，abstract修饰，可重载等。

16.什么叫类中类？主要包括哪几种类中类？请简述其特点和作用。

类种类是指在一个类的内部可以定义另一个类。这样的结构可以用来逻辑上组织彼此紧密相关的类，并且可以控制内部类的访问权限，因为内部类可以被声明为私有（private）。

类中类主要包括以下几种：

1. 静态嵌套类

特点：静态嵌套类是定义在另一个类内部的静态类，它不需要对外部类的实例的引用就可以被创建。

作用：允许嵌套类的对象独立于外部类的对象存在。常用于当嵌套类不需要访问外部类的实例字段或方法时。

1. 内部类：

特点：内部类是定义在另一个类内部的非静态类，它需要外部类的一个实例。内部类可

以访问外部类的所有成员（包括私有成员）。

作用：用来处理外部类中复杂的逻辑，也可以用于隐藏内部实现的细节，或者当某个类

只在某个外部类的上下文中有用时。

1. 局部内部类：

特点：在方法或作用域内定义的类，不可以被声明为public，private，protected或static。

这些内部类对外完全隐藏。

作用：允许你在一些需要定义的地方制造辅助类，而不必将它们作为外部类的一部分对外暴露。

1. 匿名内部类：

特点：匿名内部类是没有名称的类，它们通常用于实例化某个具体类或某个接口的子类。

作用：它们广泛用于简单的扩展或实现类并且只用使用一次的情况下，例如在GUI中

的事件监听器。

17.简述静态代码块/构造代码块/同步代码块/普通代码块的特点。

（1）静态代码块：

特点：使用static关键字定义，并且可以在类中定义多个。

作用：主要用于初始化类级别的数据（即静态变量）。静态代码块只在类加载进内存时执行一次。

（2）构造代码块：

特点：构造代码块是在类中定义的，不使用任何前缀关键字。在每次创建实例时，构造

代码块都会被执行。

作用：通常用于所有构造方法共有的初始化代码。构造代码块在每个构造方法执行前被调用。

1. 同步代码块：

特点：同步代码块是被synchronized关键字修饰的块，用于控制访问程序，确保多线

程环境下的同步。

作用：同步代码块的目的是为了锁定某个对象，在该对象的锁在同一时刻只能被一个线程获取，以此来保证线程安全。

1. 普通代码块：

特点：普通代码块是在方法体内或其他代码块内进行逻辑划分的代码集合，它们被大括

号{}包围。

作用：普通代码块的目的是提供一种逻辑单元的执行，帮助组织和分割代码，以增强可读性和维护性。它们没有特别的关键字修饰，按序执行。

18. 简述实现字节流/字符流的读写过程。

（1）字节流：

读取（Input）：创建FileInputStream对象来打开文件进行读取。使用read（）方法从流中读取数据。Read（）方法有几个不同的重载版本，可以一次读取单个字节或整个字节数组。读取完毕后，用close（）方法关闭流以释放系统资源。

写入（Output）：创建FileOutputStream对象类打开文件进行写入。使用write（）方法将数据写入文件。同样，可以写入单个字节或字节数组。写入完成后，使用close（）方法关闭流。可选的，可以调用flush（）方法关闭流以释放系统资源。

1. 字符流：

读取（Input）：创建FileInReader对象来打开文件进行读取。使用read（）方法读取字符；它可以一次读取单个字符或字符数组。读取操作完成后使用close（）方法关闭流。

写入（OutPut）：创建FileWriter对象用于写入数据到文件。使用write（）方法写入字符串或字符数组到文件中。写入完成后，同样用close（）方法关闭流，或者使用flush（）刷新写入器确保所有缓冲数据已写出。

1. 简述 TCP/UDP 网络编程的基本原理（包收发的实现过程）。

TCP网络编程基本原理：

1. 服务器端：创建ServerSocket实例：指定监听端口；调用accept（）方法：等待客户端的连接；accept（）方法返回Socket实例：表示服务其与客户端的连接；获取输入流和输出流：通过Socket实例的getInputStream（）和getOutputStream（）方法与客户端通信；数据读写：使用输入流读取数据，使用输出流发送数据；关闭连接：数据传输完毕后关闭Socket和ServerSocket。
2. 客户端：创建Socket实例：指定服务器的IP地址和端口号；获取输入流和输出流：就像服务其端一样；数据读写：通过输出流发送数据给服务器，通过输入流接受服务器响应；关闭连接：通信完成后关闭Socket。

UDP网络编程基本原理：

1. 发送数据：创建DatagramSocket实例：可以选择绑定特定端口；创建DatagramPacket：指定发送的数据，数据长度，接收方的IP地址和端口；发送数据：通过DatagramSocket的send（）方式发送DatagramPacket；关闭连接：发送完成后关闭DatagramSocket。
2. 接收数据：创建DatagramSocket实例：绑定监听的端口；创建DatagramPacket用于接受数据；接收数据：通过DatagramSocket的receive（）方法接收数据，数据会填充到DatagramPacket中；从DatagramPackrt中提取数据；接收完毕后关闭DatagramSocket。

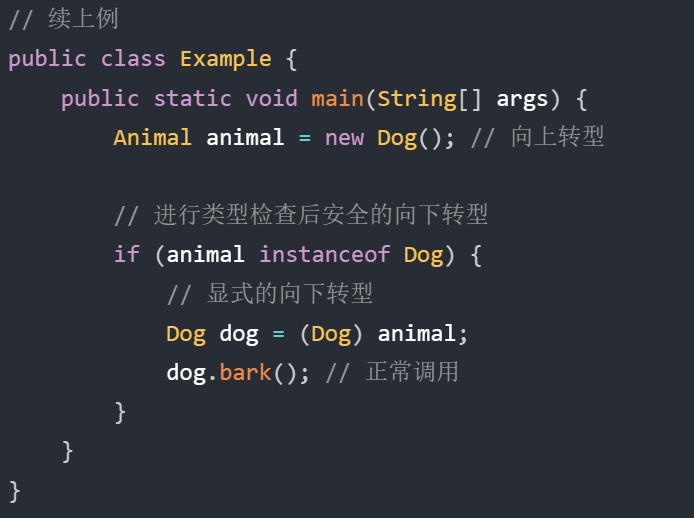
20.什么叫向上转型/向下转型？

向上转型/向下转型是与类层次结构中对象类型转换相关的两个概念。

向上转型指的是将子类的引用赋值给父类的引用。这是一种自动的、隐式的类型转换，因为子类对象自然拥有它继承的那部分父类的特征。在向上转型过程中，子类独有的方法和属性将变得不可访问，转型结果只能调用在父类中定义的方法和属性。



向下转型指的是将父类引用转换为子类类型引用。向下转型是显式的，需要强制类型转换，因为不是每个父类类型的对象都可以安全地转换为子类类型。



21.什么叫泛型？

泛型，也称类型参数化，是编程语言中一种在编写代码时允许程序员指定类、接口或方法可以操作的数据类型的功能。它使得类或方法可以用于不同的数据类型，同时保证类型的安全性。