Java 基础

1. java 面向对象有哪些特征

封装：把一个对象的状态信息（也就是属性）隐藏在对象内部，不允许外部对象直接访问对象的内部信息。但是可以提供一些可以被外界访问的方法来操作属性。

继承：让某个类型的对象获得另一个类型的对象的属性的方法。继承就是子类继承父类的特征和行为，使得子类对象（实例）具有父类的属性和方法，子类也可以对父类进行扩展，增加了代码的复用性。

多态：对于同一个行为，不同的子类对象具有不同的表现形式。多态存在的 3 个条件：1）继承；2）重写；3）父类引用指向子类对象。增加了代码的可移植性、健壮性、灵活性.

2. 访问修饰符 public、private、protected，以及不写时的区别

访问修饰符，用于控制方法和属性（成员变量）的访问权限

修饰符	同类	同包	子类	不同包/其它包
public	√	√	√	√
protected	√	√	√	×
不写（不是 default）	√	√	×	×
private	√	×	×	×

扩展：被 private 修饰的成员，只能在本类进行访问，针对 private 修饰的成员变量，如果需要被其他类使用，提供相应的操作

- 提供“get 变量名()”方法，用于获取成员变量的值，方法用 public 修饰

- 提供“set 变量名(参数)”方法，用于设置成员变量的值，方法用 public 修饰/

3. final 关键字用法

- final 修饰的类叫最终类，该类不能被继承。
- final 修饰的方法不能被重写
- final 修饰的变量叫常量，常量必须初始化，初始化之后值就不能被修改。

4. this、super 关键字

this 代表所在类的当前对象的引用（地址值），即代表当前对象。

super 代表的是父类对象的引用，this 代表的是当前对象的引用。

- super()和 this()类似,区别是，super()在子类中调用父类的构造方法，this()在本类内调用本类的其它构造方法。
- this()和 super()都指的是对象，所以，均不可以在 static 环境中使用。包括：static 变量,static 方法，static 语句块。
- super()和 this()均需放在构造方法内第一行。

5. static 关键字

1. 可以用来修饰成员变量和成员方法，该变量称为**静态变量**，该方法称为**静态方法。**
2. 无 static 修饰的成员变量或者成员方法，称为**实例变量，实例方法**
3. static 修饰的成员属于类，会存储在静态区，是随着类的加载而加载的，且只加载一次，所以只有一份，节省内存。
4. 静态代码块：只会在类加载的时候执行一次

注：被 static 修饰的变量或者方法**不属于任何一个实例对象，而是被类的实例对象所共享。**

6. 重写、重载

两者区别：

重载（Overload）：在一个类中，同名的方法有不同的参数列表（参数个数、顺序、类型不同），则视为重载，与返回类型无关。

重写（**Override**）：发生在父类子类中，若子类方法想要和父类方法构成重写关系，则它的方法名、参数列表必须与父类方法相同。另外，返回值要小于等于父类方法，抛出的异常要小于等于父类方法，访问修饰符则要大于等于父类方法。

7. ==和 equals()有什么区别?

==运算符：

- 作用于基本数据类型时，是比较两个数值是否相等；
- 作用于引用数据类型时，是比较两个对象的内存地址是否相同（引用地址）；

equals()方法：

- 没有重写时，Object 默认以 == 来实现，即比较两个对象的内存地址是否相同；
- 进行重写后，一般会按照对象的内容来进行比较，若两个对象内容相同则认为对象相等，否则认为对象不等。

equals 本质上就是 ==，只不过 String 和 Integer 等重写了 equals 方法，把引用比较改成了值比较。

8. String 创建字符串对象两种方式的区别：

- 通过构造方法创建（public　String()）

  通过 new 创建的字符串对象，每一次 new 都会申请一个内存空间，虽然内容相同，但是地址值不同

- 直接赋值方式创建（String s = “abc”;）

以" "方式给出的字符串，只要字符序列相同(顺序和大小写)，无论在程序代码中出现几次，JVM 都只会建立一个 String 对象，并在字符串池中维护

9. String 常用方法

- char charAt(int index)：返回指定索引处的字符；
- String substring(int beginIndex, int endIndex)：截取字符串
- String trim()：删除字符串两端的空格；
- int indexOf(String str)：返回子串在此字符串首次出现的索引；
- equals()：字符串比较。
- length()：返回字符串长度。
- indexOf()：返回指定字符的索引。

注：String 类由 final 修饰，所以不能被继承。

10. 字符串常量池的作用了解吗?

字符串常量池 是 JVM 为了提升性能和减少内存消耗针对字符串（String 类）专门开辟的一块区域，主要目的是为了避免字符串的重复创建

11. String str="i"与 String str=new String(“i”)一样吗?

不一样，因为内存的分配方式不一样。String str="i"的方式，Java 虚拟机会将其分配到常量池中；而 String str=new String(“i”) 则会被分到堆内存中。

12. String、StringBuffer 和 StringBuilder 的异同?

1. String 对象一旦创建，其值是不能修改的（被 final 关键字修饰），如果要修改，会重新开辟内存空间来存储修改之后的对象；而 StringBuffer 和 StringBuilder 对象的值是可以被修改的；

2. StringBuffer 几乎所有的方法都使用 synchronized 实现了同步，线程比较安全，在多线程系统中可以保证数据同步，但是效率比较低；而 StringBuilder 没有实现同步，线程不安全，在多线程系统中不能使用 StringBuilder，但是效率比较高。

3. 开发过程中如果频繁修改, 不要使用 String, 否则会造成内存空间的浪费; 当需要考虑线程安全的场景下使用 StringBuffer，如果不需要考虑线程安全，追求效率的场景下可以使用 StringBuilder。

13. JDK 和 JRE 有什么区别?

- JDK：Java Development Kit 的简称，Java 开发工具包，提供了 Java 的开发环境和运行环境。
- JRE：Java Runtime Environment 的简称，Java 运行环境，为 Java 的运行提供了所需环境。
- JVM(Java Virtual Machine)，Java 虚拟机，是 JRE 的一部分，它是整个 java 实现跨平台的最核心的部分，负责运行字节码文件

具体来说 JDK 其实包含了 JRE，同时还包含了编译 Java 源码的编译器 Javac（编译成字节码文件），还包含了很多 Java 程序调试和分析的工具。简单来说：如果你需要运行 Java 程序，只需安装 JRE 就可以了，如果你需要编写 Java 程序，需要安装 JDK。

JDK 中包含了 JRE，JRE 中包含了 JVM。

14. 抽象类相关

1. 抽象类不一定非要有抽象方法, 但是有抽象方法的类必定是抽象类
2. 抽象类中不能使用 final 修饰
3. 抽象类不能创建对象(抽象方法没有方法体,无法执行)
4. 抽象类存在的意义是为了被继承, 否则抽象类将失去意义

抽象类能使用 final 修饰吗?

不能，定义抽象类就是让其他类继承的，如果定义为 final 该类就不能被继承，这样彼此就会产生矛盾，所以 final 不能修饰抽象类。

15. 接口相关

1. 接口中的抽象方法默认会加上 public abstract 修饰, 成员变量默认会加上 public static final 修饰
2. 使用 interface 修饰, 不能实例化, 类可以实现多个接口
3. 接口可以继承接口, 并可多继承接口, 但类只能单继承

16. 抽象类和接口的区别

相同点：抽象类和接口都不能直接实例化（创建对象）

不同点：抽象类只能单继承和实现多个接口，接口可以多继承接口

抽象类中可以有构造方法，接口中不能有构造方法

关键字不同：抽象类子类使用 extends 关键字来继承抽象类，接口实现类使用关键字 implements 来实现接口；

17. 两个对象的 hashCode() 相同，则 equals() 也一定为 true，对吗?

不对，两个对象的 hashCode() 相同，equals() 不一定 true。

- 如果两个对象的 hashCode 不相同，那么这两个对象肯定是不同的两个对象
- 如果两个对象的 hashCode 相同，不代表这两个对象一定是同一个对象，也可能是两个对象
- 如果两个对象相等，那么他们的 hashCode 就一定相同

在 Java 的一些集合类的实现中，比较两个对象是否相等时，会根据上面的原则，会先调用对象 hashCode()方法得到 hashCode 进行比较，如果 hashCode 不相同，就可以直接认为这两个对象不相同，如果 hashCode 相同，那么就会进一步调用 equals()方法进行比较。而 equals()方法，就是用来最终确定两个对象是不是相等的。

18. 为什么重写 equals() 时必须重写 hashCode() 方法?

因为两个相等的对象的 hashCode 值必须是相等。也就是说如果 equals 方法判断两个对象是相等的，那这两个对象的 hashCode 值也要相等。

如果重写 equals() 时没有重写 hashCode() 方法的话就可能会导致 equals 方法判断是相等的两个对象，hashCode 值却不相等。

注：如果重写了 equals()方法，那么就要注意 hashCode()方法，一定要保证能遵守上述规则。

19. Java 数据类型及所占字节

基本数据类型有 8 种：

- 整数类型 byte、short、int、long
- 浮点类型 float、double
- 字符型 char
- 布尔型 boolean(true、false)

引用数据类型（类、接口、数组）

1 字节=8 位

数据类型	大小
1byte	8bit
1short	2byte

1int	4byte
1long	8byte
1float	4byte
1double	8byte
1char	2byte

一个中文字符占 2 个字节

注意：char 类型、一个中文字符在 java 中占 2byte，在其它语言中占 1byte，因为 Java 是 Unicode 编码，在 Unicode 编码中 1char == 2byte，在 GBK /gb2312 编码中占 2 字节，但是 utf-8 编码中占 3 字节，ASSII 编码，一个字符占 1 个字节

20. continue、break 和 return 的区别是什么?

1. continue ：指跳出当前的这一次循环，继续下一次循环。
2. break ：指跳出整个循环体，继续执行循环下面的语句。
3. return 用于跳出所在方法，结束该方法的运行。

21. 包装类相关、自动装拆箱

装箱：将基本类型用它们对应的引用类型包装起来；

拆箱：将包装类型转换为基本数据类型；

为了能够将基本数据类型当成对象操作，Java 为每一个基本数据类型都引入了对应的包装类型（wrapper class），int 的包装类就是 Integer，从 Java 5 开始引入了自动装箱/拆箱机制，使得二者可以相互转换。

Java 为每个原始类型提供了包装类型：

原始类型: boolean，char，byte，short，int，long，float，double

包装类型：Boolean，Character，Byte，Short，Integer，Long，Float，Double

包装类型的缓存机制

装箱拆箱：基本类型要转换成包装类型，需要调用包装类型的静态 valueOf()函数；

而包装类型要转换成基本类型，需要调用以基本类型开头的 Value()，比如 intValue() 。

```java
//Java 会自动帮你实现拆箱装箱
Integer x = 2;     // 装箱 相当于是 Integer x = Integer.valueOf(2)
int y = x;         // 拆箱 实际是 int y = intValue(x)
```

使用 valueOf()方法时，系统将会判断数值是否存在于缓存池中。如果在就直接返回缓存池中的对象，如果不存在会直接返回一个 new 出来的新的对象。

22. java 中 IO 流分为几种?

按照流的流向分，可以分为输入流和输出流；

按照操作单元划分，可以划分为字节流和字符流；

按照流的角色划分为节点流和处理流。

Java Io 流共涉及 40 多个类，这些类看上去很杂乱，但实际上很有规则，而且彼此之间存在非常紧密的联系，Java I0 流的 40 多个类都是从如下 4 个抽象类基类中派生出来的。

InputStream/Reader: 所有的输入流的基类，前者是字节输入流，后者是字符输入流。

OutputStream/Writer: 所有输出流的基类，前者是字节输出流，后者是字符输出流。

23. Files 的常用方法都有哪些?

Files. exists()：检测文件路径是否存在。

Files. createFile()：创建文件。

Files. createDirectory()：创建文件夹。

Files. delete()：删除一个文件或目录。

Files. copy()：复制文件。

Files. move()：移动文件。

Files. size()：查看文件个数。

Files. read()：读取文件。

Files. write()：写入文件。

24. 面向对象和面向过程的区别

解决问题的方式不同

- 面向过程把解决问题的过程拆成一个个方法，通过一个个方法的执行解决问题。
- 面向对象会先抽象出对象，然后用对象执行方法的方式解决问题。

25. 深拷贝和浅拷贝区别了解吗？（clone）

浅拷贝：拷贝基本数据类型的值以及引用数据类型的引用地址，不会复制一份引用地址所指向的对象，内部的属性指向的是同一个对象

深拷贝：拷贝基本数据类型的值，会复制一份引用地址所指向的对象，深拷贝出来的对象，内部的属性指向的不是同一个对象。

26. Java 异常体系

Java 中，所有的异常都来自顶级父类 Throwable 类，Throwable 类有两个子类 **Exception** 和 **Error**

- **Exception** :程序本身可以处理的异常，可以通过 catch 来进行捕获

  运行时异常：可以正常通过编译

1) NullPointerException 空指针异常

2) ArithmeticException 数学运算异常

3) ArrayIndexOutOfBoundsException 数组下标越界异常

4) ClassCastException 类型转换异常

5) NumberFormatException 数字格式不正确异常[]

编译异常：在编译过程，就必须处理的异常，否则代码不能通过编译

SQLException //操作数据库时，查询表可能发生异常

IOException //操作文件时，发生的异常异常

FileNotFoundException //当操作一个不存在的文件时，发生异常

ClassNotFoundException//加载类,而该类不存在时，异常

EOFException//操作文件，到文件末尾，发生异常

FileNotFoundException/当操作一个不存在的文件时，发生异常

lllegalArguementException//参数异常

- **Error**： Error 属于程序无法处理的错误，严重错误，例如 Java 虚拟机运行错误（Virtual MachineError）、虚拟机内存不够错误(OutOfMemoryError)、类定义错误（NoClassDefFoundError）等

27. JAVA 的异常捕获机制

Java 的异常捕获机制可以用 try-catch-finally 语句块来实现。通常情况下，当程序运行过程中出现异常时，程序将停止执行并输出异常信息。为了让程序能够在出现异常的情况下继续执行，可以使用 try-catch-finally 语句块来捕获异常并处理它们。

```
try {
    // 可能会产生异常的代码
} catch (ExceptionType e) {
```

```
    // 处理异常的代码
} finally {
    // 不管有没有异常都会执行的代码
}
```

Java

try 块中包含可能会产生异常的代码，如果在执行 try 块中的代码时出现了异常，就会立即跳转到 catch 块中处理异常。catch 块中的代码将根据异常类型进行处理，例如输出异常信息、记录日志、抛出新的异常等等。finally 块中的代码不管 try 块中是否发生异常都会执行，通常用于释放资源或清理工作。

如果 try 块中发生了多个异常，可以使用多个 catch 块来分别处理不同类型的异常，例如：

```
try {
    // 可能会产生异常的代码
} catch (IOException e) {
    // 处理 IO 异常的代码
} catch (SQLException e) {
    // 处理数据库异常的代码
} catch (Exception e) {
    // 处理其他类型异常的代码
} finally {
    // 不管有没有异常都会执行的代码
}
```

Java

28. 什么是反射机制?

JAVA 反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为 java 语言的反射机制。

29. 反射机制优缺点

优点： 运行期类型的判断，动态加载类，提高代码灵活度。

缺点： 性能瓶颈：反射相当于一系列解释操作，通知 JVM 要做的事情，性能比直接的 java 代码要慢很多。

30. 反射的应用场景?

Spring/Spring Boot、MyBatis 等框架设计、动态代理设计模式也采用了反射机制

31. Java 中创建对象的方式

1. new
2. clone
3. 通过反射

   使用 Class 对象的 newInstance()方法来创建该 Class 对象对应类的实例，而执行 newInstance()方法时实际上是利用默认构造器来创建该类的实例。

   通过调用构造器 Constructor 再去创建对象 Constructor.newInstance，可以选择使用指定构造器来创建实例
4. 反序列化

   序列化：指把 **Java** 对象转换为字节序列的过程；

   反序列化：指把字节序列恢复为 **Java** 对象的过程；

32. java 中 # 和 $ 的区别

#{} 表示一个占位符，可以防止 SQL 注入

${} 表示 SQL 的拼接

33. 数组和链表的区别：

（1）存储形式：数组在内存中是连续的一段空间，声明时就要确定长度。链表是一块可不连续的动态空间

（2）插入和删除元素：在数组中插入和删除元素会涉及到移动其它元素的操作；链表的节点之间通过指针相连，操作比数组效率更高。

（3）修改查找元素：数组可以通过下标直接访问，链表需要从头开始遍历查找。数组便于查询，链表便于插入删除。

34. 数组增删查的复杂度, 链表增删查的复杂度

- 数组：

  查、改，通过下标检索，O(1)

  增：最好（末尾增加）：O(1)、最坏（头部增加）：O(n)、平均复杂度：O(n)

  删：和增加一样

- 链表：

  查、改：平均复杂度 O(n)，遍历找到元素的位置，然后更改节点存放的值。

  增：平均复杂度 O(n)

  删：和增一样

35. java8 新特性：

Lamdba 表达式、Stream API

36. 内存溢出和内存泄露的区别

- 内存泄露是指程序未能正确释放不再使用的内存资源，导致内存逐渐耗尽，影响系统性能。
- 内存溢出是指程序试图分配超过可用内存大小的内存空间，导致分配失败或崩溃。

37. 栈和队列的区别

- 栈适用于需要“后进先出”的场景，如递归、回溯等。
- 队列适用于需要“先进先出”的场景，如任务调度、缓冲区等。