java基础

● 请你谈谈Java中是如何支持正则表达式操作的?

考察点：正则表达式

参考回答：

Java中的String类提供了支持正则表达式操作的方法，包括：matches()、replaceAll()、replaceFirst()、split()。此外，Java中可以用Pattern类表示正则表达式对象，它提供了丰富的API进行各种正则表达式操作，如：

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
class RegExpTest {
    public static void main(String[] args) {
        String str = "成都市(成华区)(武侯区)(高新区)";
        Pattern p = Pattern.compile(".*?(?=\\()");
        Matcher m = p.matcher(str);
        if(m.find()) {
            System.out.println(m.group());
        }
    }
}
```

● 请你简单描述一下正则表达式及其用途。

考察点：正则表达式

参考回答：

在编写处理字符串的程序时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

● 请你比较一下Java和JavaSciprt?

考察：Java&JavaScript

参考回答：

JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun Microsystems公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；而JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言。JavaScript的前身是LiveScript；而Java的前身是Oak语言。
下面对两种语言间的异同作如下比较：
- 基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言，因而它本身提供了非常丰富的内部对象

供设计人员使用。
- 解释和编译：Java的源代码在执行之前，必须经过编译。JavaScript是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行。（目前的浏览器几乎都使用了JIT（即时编译）技术来提升JavaScript的运行效率）
- 强类型变量和类型弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量是弱类型的，甚至在使用变量前可以不作声明，JavaScript的解释器在运行时检查推断其数据类型。
- 代码格式不一样。

● 请你说明一下，在Java中如何跳出当前的多重嵌套循环?

考察点：循环

参考回答：

在最外层循环前加一个标记如A，然后用break A;可以跳出多重循环。（Java中支持带标签的break和continue语句，作用有点类似于C和C++中的goto语句，但是就像要避免使用goto一样，应该避免使用带标签的break和continue，因为它不会让你的程序变得更优雅，很多时候甚至有相反的作用，所以这种语法其实不知道更好）

● 请你讲讲&和&&的区别?

考察点：运算符

参考回答：

&运算符有两种用法：(1)按位与；(2)逻辑与。&&运算符是短路与运算。逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的布尔值都是true整个表达式的值才是true。&&之所以称为短路运算是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是null而且不是空字符串，应当写为：username != null &&!username.equals("")，二者的顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的equals比较，否则会产生NullPointerException异常。

● int和Integer有什么区别?

考察点：数据类型

参考回答：

Java是一个近乎纯洁的面向对象编程语言，但是为了编程的方便还是引入了基本数据类型，但是为了能够将这些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

如：

```
class AutoUnboxingTest {
  public static void main(String[] args) {
    Integer a = new Integer(3);
    Integer b = 3;        // 将3自动装箱成Integer类型
    int c = 3;
    System.out.println(a == b);   // false 两个引用没有引用同一对象
    System.out.println(a == c);   // true a自动拆箱成int类型再和c比较
  }
}
```

● 请你说明String 和StringBuffer和StringBuilder的区别

考察点：数据类型

● 请说明String是最基本的数据类型吗?

考察点：数据类型

参考回答：

基本数据类型包括byte、int、char、long、float、double、boolean和short。
java.lang.String类是final类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用StringBuffer类。

● 请你谈谈大O符号(big-O notation)并给出不同数据结构的例子

● 请你解释什么是值传递和引用传递?

● 请你讲讲Java支持的数据类型有哪些？什么是自动拆装箱?

考察点：JAVA数据类型

参考回答：

Java语言支持的8种基本数据类型是：
byte
short
int
long
float
double
boolean
char

自动装箱是Java编译器在基本数据类型和对应的对象包装类型之间做的一个转化。比如：把int转化成Integer，double转化成Double，等等。反之就是自动拆箱。

● 请你解释为什么会出现4.0-3.6=0.40000001这种现象?

● 请你讲讲一个十进制的数在内存中是怎么存的?

● 请你说说Lamda表达式的优缺点。

● 你知道java8的新特性吗，请简单介绍一下

● 请你说明符号“==”比较的是什么?

考点：基础

参考回答：

“==”对比两个对象基于内存引用，如果两个对象的引用完全相同（指向同一个对象）时，“==”操作将返回true，否则返回false。“==”如果两边是基本类型，就是比较数值是否相等。

● 请你解释Object若不重写hashCode()的话，hashCode()如何计算出来的?

说一下排序，时间复杂度，稳定性

String拼接字符串的缺点

String是java中一个不可变的类，一旦被实例化就无法被修改，所以拼接字符串，就是生成了一个新的字符串，即原变量存储了一个新的String对象的引用。可以使用加号，或者String类中的concat方法。也可以使用StringBuilder或者StringBuffer来实现字符串拼接，这两个类的对象是可以修改的。在多线程操作时使用stringBuffer，单线程操作使用StringBuilder。

使用加号进行字符串拼接，是将 String 转成了 StringBuilder 后，使用其 `append` 方法进行处理的。

使用concat()拼接，会首先创建了一个字符数组，长度是已有字符串和待拼接字符串的长度之和，再把两个字符串的值复制到新的字符数组中，并使用这个字符数组创建一个新的 String 对象并返回。

String是基本的数据类型吗

String不是基本数据类型，java中的基本数据类型为boolean, byte,short, char, int, long, float, double八种

StringBuffer是线程安全的吗

是，Stringbuffer大部分方法上都加了synchronize加锁实现的。也因此StringBuffer执行的效率要低于StringBuilder。

两个对象equal()之后是不是相等的

equal()默认的行为是比较两个对象的引用，但在大部分java的类库中都重新实现了equals()方法，改成了比较两个对象的实际内容，例如String和Integer等。

Java锁，synchronized可以修饰静态方法吗

介绍一下wait()

使用哪个版本的jdk，有哪些特性

平时主要用java 8。java 8首要的一个新特性是支持lamda表达式和函数接口，用lambda表达式来替换单方法接口，可以把函数当成参数传递给某个方法，简化了一部分需要使用匿名类的场景（这里再去复习一下匿名类的使用场景）最简单的Lambda表达式可由逗号分隔的参数列表、**->**符号和语句块组成。函数接口是为了兼容lambda表达式产生的概念，指的是只有一个函数的接口，这样的接口可以隐式的转化成lambda表达式。Java 8 提供了一个注解**@FunctionalInterface**，来显示的说明函数式接口。

java8的另一个特性是接口中可以定义默认方法和静态方法。接口提供的默认方法会被接口的实现类继承或者重写。这样的目的是可以在不破坏现有兼容性的前提下，往接口中添加新的方法。

java8也引进了新的Date-Time Api 来改进时间，日期的处理。

java8支持重复注解

支持方法引用，通常和lambda表达式结合着使用，没用过不了解

Java list和set的区别，是否继承自Collection接口

list和set继承自Collection接口，map不是。

list和set分别是collection的两个子接口，区别为：

1. list及其实现类是可变大小的列表，适用于按照数值索引访问元素
2. Set集合无需存储，并且不可以保存重复的元素

重载和重写的区别

重写：子类在继承父类的时候，在方法名，参数列表，返回类型(除去子类中方法的返回值是父类中方法返回值的子类时)都相同的情况下，对父类的方法进行修改或重写。但要注意**子类函数的访问修饰权限不能少于父类的。**

重载：在一个类中，同名的方法如果有不同的参数列表（**参数类型不同、参数个数不同甚至是参数顺序不同**）则视为重载。同时，重载对返回类型没有要求，可以相同也可以不同

区别：重载实现的是编译时的多态性，重写实现的是运行时的多态性。重载发生在一个类中，同名的方法如果有不同的参数列表则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法有相同的参数列表，有兼容的返回类型。

面向对象的基本特征

面向对象的三个基本特征：

1. **封装**：把客观事物封装成抽象的类，并且类的属性和方法只可以被可信的类或者对象访问，对不可信的隐藏。在一个对象内部，某些代码或某些数据可以是私有的，不能被外界访问。通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分意外的改变或错误的使用了对象的私有部分。**对于这种隐藏对象属性和实现细节，仅对外公开指定方法来控制程序中属性的访问和修改，我们称之为封装。**
2. **继承**：继承是指可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

   继承的实现方式有两种：实现继承、接口继承。

   实现继承：直接使用基类公开的属性和方法，无需额外编码。

   接口继承：仅使用接口公开的属性和方法名称，需要子类实现。
3. 多态：**多态就是指一个类实例的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。**这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，它们（那些操作）可以通过相同的方式予以调用。最常见的多态就是将子类传入父类参数中，运行时调用父类方法时通过传入的子类决定具体的内部结构或行为。

   原理也很简单，父类或者接口定义的引用变量可以指向子类或者具体实现类的实例对象，由于程序调用方法是在运行期才动态绑定的，那么引用变量所指向的具体实例对象在运行期才确定。所以这个对象的方法是运行期正在内存运行的这个对象的方法而不是引用变量的类型中定义的方法。

HashMap是怎么实现的

哈希冲突：进行数据存储时，通过关键字进行hash得到的地址可能已经存储过数据了，就会出现哈希冲突。

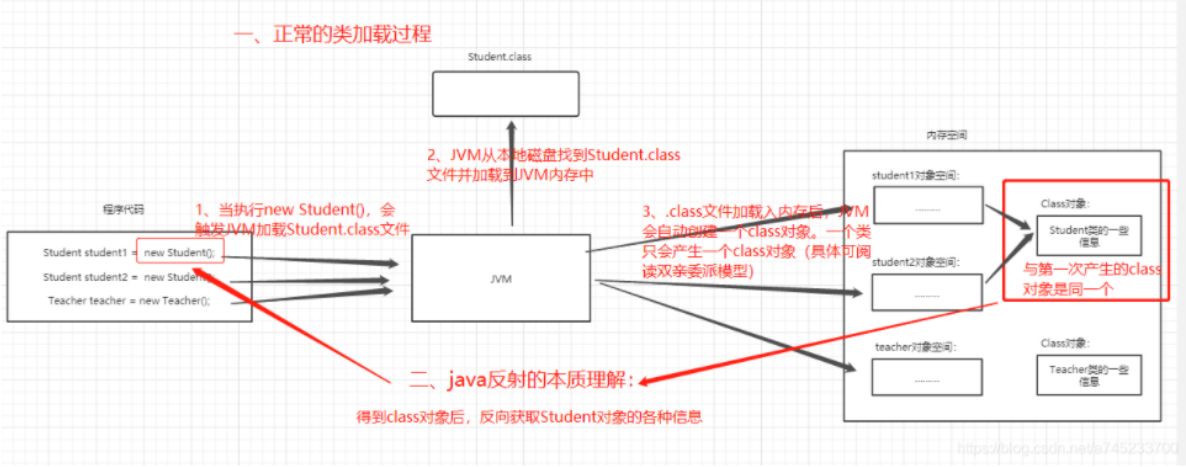
解决哈希冲突的方法有多种，开放定址法，再散列函数法，链地址法等。HashMap即采用了链地址法，也就是数组+链表的形式。数组是HashMap的主体，链表则主要为了解决哈希冲突。在jdk1.8中，HashMap采用数组+链表+红黑树来实现，当链表长度超过阈值时，将链表转化为红黑树，大大减少了查找时间。

待补充 怎么扩容？负载因子是什么

Java创建对象的不同方式

1. 使用`new`关键字（但使用new创建对象会增加耦合度，所以应该减少，啥叫增加耦合度？？？）
2. 使用反射机制创建对象

> 反射是指JVM在得到class对象之后，在通过class对象进行反编译，从而获取对象的各种信息。通过反射，可以在运行时动态的创建对象并调用其属性，不需要提前在编译时知道运行的对象是谁。



- 使用Class类的newInstance方法

```
Class heroClass = Class.forName("yunche.test.Hello");
Hello h =(Hello) heroClass.newInstance();
```

- 使用Constructor类的newInstance方法

```
Class heroClass = Class.forName("yunche.test.Hello");
Constructor constructor = heroClass.getConstructor(int.class, String.class);
Hello h =(Hello) constructor.newInstance(12, "abc");
```

Class类的`newInstance()`方法是使用该类无参的构造函数创建对象, 如果一个类没有无参的构造函数, 就不能这样创建了,可以调用Class类的getConstructor(String.class,int.class)方法获取一个指定的构造函数，然后再调用Constructor类的newInstance("张三",20)方法创建对象

3. 采用clone

   使用clone创建对象需要有一个已经分配了内存的源对象，创建新对象时，需要首先分配一个和源对象一样大的内存空间。调用clone（）方法需要实现Cloneable接口。

4. 采用序列化机制

深拷贝与浅拷贝

java中的数据类型分为基本数据类型和引用数据类型，对于这两种数据类型，在进行赋值操作和传递参数或者返回值时，会有值传递和引用传递的差别。

1. **浅拷贝**：对于一个对象中是基本数据类型的成员变量，浅拷贝直接进行值传递，即将该属性值复制一份给新的对象，两个对象间的修改不会互相影响；对于引用类型的成员变量，进行的是引用传递，两个对象的该成员指向同一个实例，两个对象的修改会互相影响。

方法1：通过一个现有的对象创建出与该对象属性相同的新的对象

```
Age a=new Age(20);
Person p1=new Person(a,"摇头耶稣");
Person p2=new Person(p1);
System.out.println("p1是"+p1);
System.out.println("p2是"+p2);
//修改p1的各属性值，观察p2的各属性值是否跟随变化
p1.setName("小傻瓜");
a.setAge(99);
System.out.println("修改后的p1是"+p1);
System.out.println("修改后的p2是"+p2);
```

输出为：

p1是摇头耶稣 20
p2是摇头耶稣 20
修改后的p1是小傻瓜 99
修改后的p2是摇头耶稣 99

注意，这里对Person类选择了两个具有代表性的属性值：一个是引用传递类型；另一个是字符串类型（属于引用数据类型，但其值无法修改，所以可看做常量），因此不是更改了字符串的值，而是修改了引用指向的字符串。

方法2：通过重写clone()方法进行浅拷贝

重写Cloneable接口，重写clone()方法，通过super.clone()方法调用object类中的原clone方法，因为object类中的clone()方法进行的就是浅拷贝，但被protected修饰，无法直接使用

代码略

2. **深拷贝**：首先介绍对象图的概念。设想一下，一个类有一个对象，其成员变量中又有一个对象，该对象指向另一个对象，另一个对象又指向另一个对象，直到一个确定的实例。这就形成了对象图。那么，对于深拷贝来说，不仅要复制对象的所有基本数据类型的成员变量值，还要为所有引用数据类型的成员变量申请存储空间，并复制每个引用数据类型成员变量所引用的对象，直到该对象可达的所有对象。也就是说，**对象进行深拷贝要对整个对象图进行拷贝。**

方法1：通过重写clone()方法进行深拷贝

反射机制实现对数据库数据的增、查例子

java的不同访问等级