1.为什么要有包装类(或封装类)

因为Java的某些特性和API，例如泛型和集合，不支持基本数据类型，所以必须把基本数据类型转为包装类的对象。

2.基本数据类型与对应的包装类：

byte->Byte

short->Short

int->Integer

long->Long

float->Float

double->Double

char->Character

boolean->Boolean

3.需要掌握的类型间的转换：

（1）基本数据类型与包装类对象之间的转换

装箱：把基本数据类型转为包装类对象

```
Integer i = new Integer(10);//手动装箱
Integer i = 10;//自动装箱
```

拆箱：把包装类的对象转为基本数据类型

```
Integer i = new Integer(10);
int a = i.intValue();//手动拆箱
int a = i;//自动拆箱
```

（2）可以使用包装类的API实现字符串与基本数据类型之间的转换

把字符串转为基本数据类型：

```
String str = "123";
int i = Integer.parseInt(str);
int i = Integer.valueOf(str);
```

把基本数据类型转为字符串：

```
int a = 10;
String str = a+"";    //任意类型与字符串进行拼接的结果就是字符串
```

应用场景举例：

（1）基本数据类型与包装类之间的转换，主要在相关API和算术运算时，来回转换

（2）字符串与基本数据类型之间的转换，web和ee项目时，从网页接收数据传到服务器端，要做数据类型转换。

4、包装类对象的缓存对象

Byte,Short,Integer,Long：-128~127

Float,Double：没有

Character：0~127

Boolean：true,false

一、java.lang.String类：

1.特点：

（1）字符串对象不可变

（2）String类型是final修饰的，不能被继承

2.内存的存储结构：

（1）JDK1.9之前：char[]

（2）JDK1.9：byte[]

3.常用方法：

（1）length()：求字符串的长度
（2）trim()：去掉前后空格
（3）equals()：比较两个字符串是否“相等”，比较字符串的内容
（4）equalsIgnoreCase()：比较两个字符串是否“相等”，比较字符串的内容，忽略大小写
（5）toUpperCase()：转为大写
（6）toLowerCase()：转为小写
（7）concat()：等价于“+”，拼接
（8）contains()：是否包含xx
（9）toCharArray()：转为字符数组
（10）charAt()：返回某个索引位置的字符
（11）intern()：把结果放到常量池中
（12）startsWith()：判断字符串是否以xx开头
（13）endsWith()：判断字符串是否以xx结尾
（14）indexOf()：查找某个字符,子串在当前字符串的索引，从左往右查找

4.2 与字节数组之间的转换：

编码：byte[] getBytes()
解码：new String(byte[])

4.3 与字符数组之间的转换：

转成字符数组：char[] toCharArray()

转为字符串：new String(char[])

5.2 String s = new String("hello");在内存中创建了几个对象？

两个，一个在常量池，一个在堆中

```
* 字符序列：String,StringBuffer,StringBuilder
* String类型的对象不可变字符序列。所以又配备了另一个类StringBuffer，它是可变的字符序列。
*
* StringBuffer：创建对象必须用new
* StringBuffer的拼接不能直接用+，可以用append
```

```
* StringBuffer：创建对象必须用new
* StringBuffer的拼接不能直接用+，可以用append
*
* 常用方法：
* （1）append系列：用于追加xx
* （2）insert系列：插入xx
* （3）delete系列：删除
* （4）reverse系列
```

```
* StringBuffer：又称为字符串缓冲区，内部用char[]数组存储。
*  数组的长度是不可变。
* （1）char[]的初始化的长度是多少？
*       StringBuffer():默认是16
*       StringBuffer(String str)：默认长度是str.length+16
*       StringBuffer(int capacity) ：默认长度是由capacity指定
* （2）char[]如果存满了，怎么办？
*       先把value的数组扩大为value.length*2+2，如果还不够，就按实际需要的来。
```

```
* StringBuilder：JDK1.5
*   StringBuilder与StringBuffer的API完全兼容，但是StringBuilder不保证同步。
*  换句话说：StringBuffer是线程安全的，StringBuilder是线程不安全。
*  即当多线程来共同使用同一个StringBuffer的对象时，是安全的，
*  即当多线程来共同使用同一个StringBuilder的对象时，是不安全的。
```

```
面试题：StringBuilder与StringBuffer、String三个的区别？
String：对象不可变
StringBuilder与StringBuffer：对象可变
StringBuffer是线程安全的，StringBuilder是线程不安全。
```

```
* System有三个常量对象：
* System.in：InputStream
* System.out：PrintStream
* System.err：PrintStream
*
* System.currentTimeMillis()：当前时间与协调世界时 1970 年 1 月 1 日午夜之间的时间差（以毫秒为单位测量）。
* System.arraycopy(src,srcPost,dest,destPost,int len)
*       第一个参数：被复制（移动）数组
*       第二个参数：从哪个位置开始复制（移动）
*       第三个参数：目标数组
*       第四个参数：目标的起始位置
*       第五个参数：要复制（移动）元素个数
```

System有三个常量对象：
System.in：InputStream
System.out：PrintStream
System.err：PrintStream

System.currentTimeMillis()：当前时间与协调世界时 1970 年 1 月 1 日午夜之间的时间差（以毫秒为单位测量）。
System.arraycopy(src,srcPost,dest,destPost,int len)
第一个参数：被复制（移动）数组
第二个参数：从哪个位置开始复制（移动）
第三个参数：目标数组
第四个参数：目标的起始位置
第五个参数：要复制（移动）元素个数

System.gc()：通知垃圾回收器来回收垃圾。但是，不是一调用，里面就来。

Runtime：代表运行环境。
Runtime对象.gc()
long maxMemory()
long freeMemory()

Math：和数学运算相关
如初等指数、对数、平方根和三角函数。

常量：
1、PI
2、求平方根：sqrt(x)
3、求几次方：pow(x,y)，求x的y次方
4、round(x)
ceil(x)
floor(x)

5、max(x,y)
min(x,y)

6、 随机数：Math.random() -》[0,1)

Java中在设计很多方法时，右边界基本上不包括。

java.util.Random类：专门用来产生随机数

6、 随机数：Math.random() -》[0,1)

Java中在设计很多方法时，右边界基本上不包括。

java.util.Random类：专门用来产生随机数
1、double nextDouble() :[0,1)
2、int nextInt()：遍布所有int范围
3、int nextInt(int n)：[0,n)

java.math包：
1、BigInteger：
大整数
2、BigDecimal：
不可变的、任意精度的有符号十进制数。

面试题：int,Integer,BigInteger的区别？
int：基本数据类型
Integer：包装类
BigInteger：不可变的任意精度的整数。

double,Double,BigDecimal的区别？
...

计算方面：
基本数据类型：直接+,-,*,/。。。
包装类：拆箱再计算
BigDecimal，BigInteger：通过方法

日期时间API：
第一代：java.util.Date
(1)年份：1900
第二代：java.util.Calendar

第三代：JDK1.8之后引入了新的日期时间的API ，例如：LocalDate，LocalTime等
(1)老版的日期时间对象没有设计为不可变对象
每一个日期时间点，都应该用一个对象表示。不同的日期时间点应该用不同的对象表示。
(2)老版的日期时间API没有考虑闰秒
(3)关于日期格式化等API只支持Date类型，不支持Calendar

```
* 第一代：java.util.Date
* 1、new Date()
* 2、long getTime()
* 3、new Date(毫秒)
*
*
* java.sql.Date是在JDBC中使用。
*
* 第二代：java.util.Calendar
```

```
* 第二代：java.util.Calendar
*   子类：GregorianCalendar
*
* 1、Calendar c = Calendar.getInstance();
* 2、get(常量名)获取具体的某个时间值
*
* 关于日期时间格式化：
*
```

```
* 第二代：java.util.Calendar
*   子类：GregorianCalendar
*
* 1、Calendar c = Calendar.getInstance();
* 2、get(常量名)获取具体的某个时间值
*
* 关于日期时间格式化和解析：
*  java.text.DateFormat：
*  java.text.SimpleDateFormat：
```

```
* LocalDate：只能表示日期，例如：生日，入职日期
* LocalTime：只能表示时间
* LocalDateTime：可以表示日期加时间，例如：用户注册日期时间，最后的登录日期时间，订单日期时间
*
* 1、now()
* 2、
*/
```

```
* LocalTime：只能表示时间
* LocalDateTime：可以表示日期加时间，例如：用户注册日期时间，最后的登录日期时间，订单日期时间
*
* 1、now()
* 2、of()
* 3、getYear()
*    getXxx()
* 4、withXxx()：修改
* 5、plusXxx()：加
* 6、minusXxx()：减
* 7、isLeapYear()：是否是闰年
```

```
/*
 * Duration：时间间隔
 * Period：日期间隔
 *    between(x,y)
 */
```

容器：用来装数据的

1、数组

优点：可以根据索引快速的定位到某个元素，访问速度非常快

缺点：长度是固定的，如果数组满是了，需要考虑扩容，并且删除和插入元素时，需要移动元素。

需要另一个变量，例如：total，来辅助记录实际有效元素的个数。

2、集合

集合新设计的一组容器，具有各种特点。

数据结构

栈、队列、链表、堆、图、树、哈希表...

无论多复杂，都是从以下两种物理结构的基础上构建出来的：

（1）数组

在内存中需要开辟连续的存储空间。有可能有很多空闲的空间

元素类型：就是数据的类型

（2）链式

在内存中不需要连续的空间。不会有空闲空间。

元素类型：结点类型

结点类型：

单向链表

```
class Node{
    Object data;
    Node next;
}
```

双向链表

```
class Node{
    Node previous;
    Object data;
    Node next;
}
```

树

```
class Node{
    Node parent;
    Object data;
    Node left;
    Node right;
}
....
```