包装类和字符串

回顾

今天任务

```
1.基本数据类型与其包装类
        1.1 基本数据类型复习
        1.2 基本数据类型所对应的包装类
        1.3 数据的装箱和拆箱
        1.4 基本类型与字符串之间的相互转换
2.字符串
        2.1 String类
        2.2 StringBuffer类
        2.3 StringBuilder类
        2.4 正则表达式
```

教学目标

```
1.了解基本数据类型对应的对象包装类
2.掌握数据的装箱和拆箱
3.掌握基本数据类型和字符串之间的相互转换
4.掌握String类中的常用方法
5.掌握StringBuffer类中的常用方法
6.了解StringBuffer和StringBuilder之间的区别
7.了解正则表达式以及常见的用法
```

第一节 基本数据类型与其包装类

1.1 基本数据类型复习

> byte(1字节)、short(2字节)、int(4字节)、long(8字节)、float(4字节)、double(5字节)、char(2字节)、boolean(1字节)

1.2 基本数据类型所对应的包装类

> byte Byte short Short int Integer long Long float Float double Double char Character boolean Boolean

1.3 数据的装箱和拆箱

> 拆箱：将包装类型数据转换成基本类型数据 装箱：将基本类型数据包装成包装类型数据 装箱和拆箱的方法：
> 装箱：使用包装类中的构造方法，或静态valueOf方法

```
        int a = 5;
        Integer i = new Integer(a);
```

```
        double b = 3.0;
        Double d = Double.valueOf(b);
```

> 拆箱：使用包装类中的xxValue方法

```
        Integer i = new Integer(5);
        int a = i.intValue();
```

> 自动装箱和自动装箱：jdk1.5之后新增的功能
>
> 自动装箱：可以直接将基本类型数据赋值给包装类对象

```
        int a = 5;
        Integer i = a;
```

> 自动拆箱：直接将包装类对象数据赋值给基本类型变量

```
        Integer i = new Integer(5);
        int a = i;
```

1.4 基本类型与字符串之间的相互转换

> 基本类型转字符串：
>
> 1）字符串连接符：任何基本类型数据与字符串链接都变成字符串形式

```
        int a = 5;
        //将基本类型变量与一个空字符串链接
        String str = a + "";
```

> 2）String类中的valueOf方法：

```
        boolean boo = true;
        String str = String.valueOf(boo);
```

> 字符串转基本类型：
>
> 1）包装类中的parseXx方法：

注意：

i. 字符串不能直接转成字符类型，需要使用String类中的charAt方法去字符串中的第一个字符

ii. 若字符串转数值类型时，若字符串中存在不能表示数值的字符时，抛出

java.lang.NumberFormatException异常

iii. 字符串转布尔类型时，当且仅当字符串是“true”时，结果为true，否则其他任意字符串转布尔类型结

果都是false

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

> 2）可以使用包装类中的valueOf(String str)方法：Character类中没有valueOf(String str)方法 valueOf方法返回值类型时包装类对象，所以在jdk1.5之前不能使用此方法实现字符处串转基本类型

```
        String str = "ture";
        boolean boo = Boolean.valueOf(str);//结果为false
```

第二节 字符串

> 字符串常量：
>
> 字符串数据是常量，存储在常量池中，常量池中不允许存储相同的数据，字符串可以直接将数据赋值给对象引用.

```
    String str = "123abc";
```

2.1 String类

> 常用构造方法

方法名	描述
String()	创建出一个字符串对象，此字符串中没有任何字符，空字符串
String(byte[] bytes, String charsetname)	通过使用指定的charset解码指定的 byte 数组，构造一个新的 String
String(String str)	初始化一个新创建的String对象，使其表示一个与参数相同的字符序列

```
        //创建一个字符串对象
        String s = new String("abcdefgghjklll");
```

> 常用的成员方法：
>
> 2.1.1 获取字符串的长度：

```
        /*
         int length()           返回此字符串的长度。
        */
        int len = s.length();//获取字符串s的长度
```

> 2.1.2 获取某个字符或者字符串在原字符串中第一次出现的位置

```
/*
 int indexOf(int ch)
        返回指定字符在此字符串中第一次出现处的索引。
 int indexOf(int ch, int fromIndex)
        返回在此字符串中第一次出现指定字符处的索引，从指定的索引开始搜索。
 int indexOf(String str)
        返回指定子字符串在此字符串中第一次出现处的索引。
 int indexOf(String str, int fromIndex)
        返回指定子字符串在此字符串中第一次出现处的索引，从指定的索引开始。
*/
int index1 = s.indexOf('g');//获取'g'字符在s字符串中第一次出现的索引

//注意：返回的是在源字符串中的索引
int index2 = s.indexOf('g',4);//获取'g'字符在s字符串中从索引为4开始第一次出现的索引

//注意：查找字符串时，返回的是第一个字母的下标
int index3 = s.indexOf("abc");//获取"abc"字符串在s字符串中第一次出现的索引

//注意：查询没有结果时则返回-1
int index4 = s.indexOf("abc",4);//获取"abc"字符串在s字符串中从索引为4开始第一次出现的索引
```

2.1.3 获取某个字符或者字符串在原字符串中最后一次出现的位置

```
/*
int lastIndexOf(int ch)
    返回指定字符在此字符串中最后一次出现处的索引。
int lastIndexOf(int ch, int fromIndex)
    返回指定字符在此字符串中最后一次出现处的索引，从指定的索引处开始进行反向搜索。
int lastIndexOf(String str)
    返回指定子字符串在此字符串中最右边出现处的索引。
int lastIndexOf(String str, int fromIndex)
    返回指定子字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索。
*/

//注意：从左往右进行查询，获取到的索引仍然是在原字符串中的索引
int index5 = s.lastIndexOf('g');//获取'g'字符在s字符串中最后一次出现的位置
```

2.1.4 获取某个位置上的字符

```
/*
char charAt(int index)            返回指定索引处的 char 值。
*/
char ch = s.charAt(6);//获取s字符串中索引为6的字符
```

2.1.5 判断字符串中是否包含某个子字符串

```
/*
 boolean contains(CharSequence s)     当且仅当此字符串包含指定的 char 值序列时，返回 true。
*/
//注意：判断包含的内容必须是连续的
boolean b1 = s.contains("hello");//判断s字符串是否包含"hello"字符串
```

2.1.6 判断字符串中是否有内容

```
/*
 boolean isEmpty()            当且仅当 length() 为 0 时返回 true。
*/
boolean b2 = s.isEmpty();//判断s字符串是否是空字符串
```

2.1.7 判断字符串是否是以某个前缀开始的

```
/*
 boolean startsWith(String prefix)          测试此字符串是否以指定的前缀开始。
*/
//注意：区分startsWith和contains
boolean b3 = s.startsWith("hello");//判断s字符串是否以"hello"开头
```

2.1.8 判断字符串是否是以某个后缀开始的

```
/*
 boolean endsWith(String suffix)
  测试此字符串是否以指定的后缀结束。
*/
boolean b4 = s.endsWith("hello");//判断s字符串是否以"hello"结尾
```

2.1.9 判断字符串的内容是否相等

```
/*
 boolean equals(Object anObject)                    将此字符串与指定的对象比较。
*/
String str1 = "hello world";
boolean b5 = s.equals(str1);//判断str1是否与s相同
```

2.1.10 忽略大小写进行比较

```
/*
 boolean equalsIgnoreCase(String anotherString)
 将此 String 与另一个 String 比较，不考虑大小写。
*/
boolean b6 = "Abc".equalsIgnoreCase("abc");//true
boolean b7 = "Abc".equals("abc");//false
```

2.1.11 替换

```
/*
 String replace(char oldChar, char newChar)
 返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的。
String replace(CharSequence target, CharSequence replacement)
 使用指定的字面值替换序列替换此字符串所有匹配字面值目标序列的子字符串。
*/
String s = "hello java";
//注意：会替换原字符串中所有的指定字符
String s1 = s.replace('a','k');
System.out.println(s1);

//注意：将要替换的字符串可以和被替换的字符串长度不相等，当做一个整体被替换掉
String s2 = s.replace("java","php");
System.out.println(s2);
```

2.1.12 截取

```
/*
 String substring(int beginIndex)
 返回一个新的字符串，它是此字符串的一个子字符串。
String substring(int beginIndex, int endIndex)
 返回一个新字符串，它是此字符串的一个子字符串。
*/
//从指定的下标开始截取后半部分
String str2 = str1.substring(3);
System.out.println(str2);

//包头不包尾：截取指定区间的子字符串
String str3 = str1.substring(3,9);
System.out.println(str3);
```

2.1.13 去除前面和尾部的空格

```
/*
 String trim()
 返回字符串的副本，忽略前导空白和尾部空白。
*/
String string1 = "   hello    hello  ";
String string2 = string1.trim();
System.out.println(string2);
```

2.1.14 格式化字符串:将字符串按照指定的格式输出

```
/*
static String format(String format, Object... args)
 使用指定的格式字符串和参数返回一个格式化字符串。

%f   float
%d    整型
%s     对象
%c      char
%b       boolean
*/
//注意：可以保留小数点后几位
String string3 = String.format("%.2f--%b--%s",10.23766f,true,"hello");
System.out.println(string3);
System.out.println("hello" + 10.23766f + true);
```

2.1.15 比较

```
        /*
         int compareTo(String anotherString)
         按字典顺序比较两个字符串。
        int compareToIgnoreCase(String str)
         按字典顺序比较两个字符串，不考虑大小写。
        */
        /*
        如果按字典顺序此 String 对象位于参数字符串之前，则比较结果为一个负整数。
        如果按字典顺序此 String 对象位于参数字符串之后，则比较结果为一个正整数。
        如果这两个字符串相等，则结果为 0
        */
        int num1 = "abc".compareTo("def");
        System.out.println(num1);//-3

        int num2 = "def".compareTo("abc");
        System.out.println(num2);//3

        int num3 = "abc".compareTo("abc");
        System.out.println(num3);
```

2.1.16 拼接

```
        /*
         String concat(String str)
         将指定字符串连接到此字符串的结尾。
        */

        //注意：在String类中，但凡返回值是String类型的方法，生成的都是一个新的字符串，跟原来的字符串没有关系
        String newStr = str1.concat("hello");
        System.out.println(str1);//welcome to china
        System.out.println(newStr);//welcome to chinahello
```

2.1.17 内存中的字符串

```java
class StringUsageDemo01
{
    public static void main(String[] args)
    {
        //s1表示引用，存储在栈空间中，但是，“hello”存储在常量池中
        String s1 = "hello";
        String s2 = "hello";
        //注意1：使用一个字符串常量定义两个不同的变量，这时两个变量其实在内存中是同一块内存空间
        //原因：两个变量都拷贝了字符串常量的地址
        System.out.println(s1 == s2);//true
        System.out.println(s1.equals(s2));//true

        //注意2：但凡遇到new关键字，表示开辟了不同的空间
        //s3和s4分别指向了两个不同的有效空间
        String s3 = new String("hello");
        String s4 = new String("hello");
        System.out.println(s3 == s4);//false
        System.out.println(s3.equals(s4));//true

        System.out.println(s1 == s3);//false
        System.out.println(s1.equals(s3));//true

        /*
        String s1 = "hello";          只有一个对象，是“hello”
        String s3 = new String("hello");     两个对象，一个是“hello”,另外一个就是new出现的对象
        */

        //字符串是一个特殊的对象，一旦被初始化之后将不能发生改变
        //注意3：不能发生改变指的是真正的对象【字符串常量对象和new出现的对象】
        s1 = "java";

        /*
        String str = "abc";
        等价于char[] arr = {'a','b','c'};

        a.字符串的底层是一个字符数组，默认的修饰符为final
        b.访问权限修饰符是private，私有的，没有提供对外的可以修改数组的方法，所以数组中的元素不能发生改
变
        */
    }
}
```

2.1.18 练习

```
class PracticeDemo
{
    public static void main(String[] args)
    {
        /*
        需求：
        已知String str = "this is a text";
        1.将str中的单词单独获取出来
        2.将str中的text替换为practice
        3.在text前面插入一个easy
        4.将每个单词的首字母改为大写
        */
        String str = "this is a text";

        //1.切割
        String newString = "";
        String[] arr = str.split(" ");
        for(String s:arr) {
            System.out.println(s);

            //获取每个单词的首字母
            char ch = s.charAt(0);
            //转化为大写字母
            char newCh = (char)(ch -32);

            //String ss = s.replace(s.charAt(0),newCh);
            StringBuffer sb = new StringBuffer(s);
            sb.setCharAt(0,newCh);

            String newStr = sb.toString();
            newStr += " ";
            newString += newStr;
        }
        System.out.println(newString);

        //2.替换
        String newStr = str.replace("text","practice");
        System.out.println(newStr);

        //3.插入
        StringBuffer buf = new StringBuffer(str);
        buf.insert(buf.length() - "text".length(),"easy ");
        System.out.println(buf);
    }
}
```

2.2 StringBuffer类

> 字符串缓冲区：使用缓冲区操作字符串要比直接操作字符串效率高
>
> StringBuffer中的常用构造方法：

StringBuffer(String str)	构造一个字符串缓冲区，并将其内容初始化为指定的字符串内容。
StringBuffer()	构造一个其中不带字符的字符串缓冲区，其初始容量为 16 个字符。
StringBuffer(CharSequence seq)	构造一个字符串缓冲区，它包含与指定的 `CharSequence` 相同的字符。
StringBuffer(int capacity)	构造一个不带字符，但具有指定初始容量的字符串缓冲区。

> 常用的成员方法
>
> 2.2.1 增加

```
        /*
        StringBuffer append(String str)
       StringBuffer insert(int offset, String str)
        */
        StringBuffer sb2  = sb1.append("hello");

        //区别于String类：面盆理论
        System.out.println(sb1);
        System.out.println(sb2);
        System.out.println(sb1 == sb2);

        sb1.append("java");
        System.out.println(sb1);
        System.out.println(sb2);

        //方法链
        sb1.append("java").append("java").append("java").append("java");
        System.out.println(sb1);
        System.out.println(sb2);

        //插入
        sb1.insert(2,"hhhhhhhh");
        System.out.println(sb1);
```

2.2.2 删除

```
        /*
         StringBuffer delete(int start, int end)
        StringBuffer deleteCharAt(int index)
        */

        //删除指定区间的字符串
        sb1.delete(2,3);
        System.out.println(sb1);

        //删除指定位置上的字符
        sb1.deleteCharAt(0);
        System.out.println(sb1);
```

2.2.3 修改

```
        /*
         StringBuffer replace(int start, int end, String str)
         void setCharAt(int index, char ch)
        */
        //替换指定区间的字符串
        sb1.replace(2,5,"nnnnnn");
        System.out.println(sb1);

        //替换指定位置上的字符
        sb1.setCharAt(0,'x');
        System.out.println(sb1);
```

> 2.2.4 获取

```
        //和String类中的用法相同
        /*
        indexOf
        lastIndexOf
        charAt
        length
        substring
        */
```

> 2.2.5 反转

```
        // StringBuffer reverse()
        StringBuffer sb3 = new StringBuffer("my name is zhansan");
        sb3.reverse();
        System.out.println(sb3);
```

2.3 StringBuilder类

> StringBuilder类也是字符串缓冲区，类中的方法与StringBuffer类中的方法使用方法一样，区别在于StringBuilder类中的方法都是线程安全的，而StringBuffer类中的方法都是非线程安全的

2.4 正则表达式

> 正则表达式就是一个验证字符串格式是否满足要求的字符串

字符类

x	字符x
[^abc]	任何字符，除了 a、b 或 c（否定）
[a-zA-Z]	a 到 z 或 A 到 Z，两头的字母包括在内（范围）
[a-d[m-p]]	a 到 d 或 m 到 p：[a-dm-p]（并集）
[a-z&&[def]]	d、e 或 f（交集）
[a-z&&[^bc]]	a 到 z，除了 b 和 c：[ad-z]（减去）
[a-z&&[^m-p]]	a 到 z，而非 m 到 p：[a-lq-z]（减去）

> 预定义字符类
>
> . 任意字符（与行结束符可能匹配也可能不匹配）
>
> \d 数字：[0-9]
>
> \w 单词字符：[a-zA-Z_0-9]

> 边界匹配器
>
> ^ 行开头
>
> $ 行结尾

> 数量：
>
> X? 一次或0次
>
> X* 0次或多次(包括1次)
>
> X+ 一次或多次
>
> X{n} 恰好n次
>
> X{n, } 至少n次
>
> X{n,m} 至少n次，不超过m次

> 使用String类中的matches方法验证字符串格式：
>
> 判断QQ邮箱格式是否正确：

```
String regex = "^[1-9][0-9]{5,12}@[qQ][.][cC][oO][mM]$";
String email = "1234567@qq.com";
boolean boo = email.matches(regex);
```

> 使用split方法将字符串按照指定的标记拆分：

```
        String regex = "[ ,]";
        String str = "Hello I am LiLei,welcome to China";
        String[] strs = str.split(regex);
        System.out.println(Arrays.toString(strs));
```

总结

课前默写

```
1.设计一个成员内部类，有fun方法
2.关键字
3.final关键字用法
4.static关键字用法：只能修饰类成员
```

作业

```
1.输出今天是今年中的第几天，第几周
2. 把当前的时间以 2016年9月5日 18:00:00 的格式输出
3. 从控制台录入一个字符串，判断字符'a'在该字符串中出现的次数
4. 完成猜拳游戏
   请输入你的选择:
   1)石头
   2)剪刀
   3)布
   你的选择是【布】，电脑的选择是【石头】
   恭喜你获得了胜利!
```

面试题

```
1.String str = new String("abc");创建出了几个字符串对象;
2.String s1 = "abc";
    String s2 = new String("abc");
    String s3 = "a";
    String s4 = "bc";
    String s5 = s3 + s4;
    boolean b1 = s1==s2;
    boolean b2 = s1.equals(s2);
    boolean b3 = s1==s5;
    boolean b4 = s1.equals(s5);
    b1,b2,b3,b4的值分别是多少?
```