实用类

- 枚举 Enum

是一种特殊的数据类型，是一个类同时又比普通类多了一些约束，因为有约束，所以枚举具有简洁，安全，方便等特点。枚举的值被约束到一个特定的范围之中，只能取该范围以内的值。

在实际开发中，如果需要描述值有其特定的范围的数据时，比如性别，一年四季，周一到周天，可以使用枚举类型来描述。

枚举是由一组常量组成的类型，指定了一个区间，我们只能从该区间内取值。

枚举的定义和类很相似，使用enum关键字来描述，基本语法：

public enum 枚举名{

值1，值2，值3...

}

```
package com.southwind.test;

public enum WeekEnum {
    MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY,SUNDAY;
}
```

Java在编译期间会自动生成一个WeekEnum，并且是继承java.lang.Enum，同时被final修饰，表示该类不可被继承。

同时还生成了7个WeekEnum的实例对象分别对应枚举中定义的7个日期。

MONDAY的值是一个WeekEnum对应的抽象类的实例对象

TUESDAY的值也是一个WeekEnum对应的抽象类的实例对象

...

```
final class WeekEnum extends Enum{
    public static final WeekEnum MONDAY;
    public static final WeekEnum TUESDAY;
    public static final WeekEnum WEDNESDAY;
    public static final WeekEnum THURSDAY;
    public static final WeekEnum FRIDAY;
    public static final WeekEnum SATURDAY;
    public static final WeekEnum SUNDAY;
    private static final WeekEnum $VALUES[];

    public WeekEnum(String s,int i){
        super(s,i);
    }
```

```
    static{
        MONDAY = new WeekEnum("MONDAY",0);
        TUESDAY = new WeekEnum("TUESDAY",1);
        WEDNESDAY = new WeekEnum("WEDNESDAY",2);
        THURSDAY = new WeekEnum("THURSDAY",3);
        FRIDAY = new WeekEnum("FRIDAY",4);
        SATURDAY = new WeekEnum("SATURDAY",5);
        SUNDAY = new WeekEnum("SUNDAY",6);
        $VALUES = (new WeekEnum[]
{MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY,SUNDAY});
    }

    public static WeekEnum[] values(){
        return (WeekEnum[])$VALUES.clone();
    }

    public static WeekEnum valueOf(String s){
        return (WeekEnum)Enum.valueOf("com/southwind/test/WeekEnum",s);
    }

}
```

- Math

Math类为开发者提供了一系列的数学相关操作的静态方法，同时还提供了两个静态常量E（自然对数的底数）和PI（圆周率）。

```
package com.southwind.test;

public class MathTest {
    public static void main(String[] args) {
        /*
         * 常量E和PI
         */
        System.out.println("常量E: "+Math.E);
        System.out.println("常量PI: "+Math.PI);

        /*
         * 求平方根
         */
        System.out.println("5的平方根: "+Math.sqrt(9));

        /*
         * 求立方根
         */
        System.out.println("8的立方根: "+Math.cbrt(8));
```

```
        /*
         * 求x的y次方
         */
        System.out.println("2的3次方："+Math.pow(2, 3));

        /*
         * 求num1和num2中的较大值
         */
        System.out.println(Math.max(6, 5.5));

        /*
         * 求num1和num2中的较小值
         */
        System.out.println(Math.min(3, 2));

        /*
         * 求绝对值
         */
        System.out.println(Math.abs(-6));

        /*
         * 求大于num的最小整数
         */
        System.out.println(Math.ceil(4.1));

        /*
         * 求小于num的最大整数
         */
        System.out.println(Math.floor(4.999));

        /*
         * 四舍五入
         */
        System.out.println(Math.rint(5.6));

        /*
         * 四舍五入
         * num为float类型时返回int类型
         * num为double类型时返回long类型
         */
        System.out.println(Math.round(5.6f));
        System.out.println(Math.round(5.6));

        /*
         * 生成一个大于等于0.0且小于1.0的随机数
         */
        System.out.println(Math.random());
    }
}
```

- Random

```
package com.southwind.test;

import java.util.Random;

public class RandomTest {
    public static void main(String[] args) {
        /*
         * 无参构造
         */
        Random random = new Random();
        for (int i = 1; i <=10; i++) {
            /*
             * 随机生成一个boolean的值
             */
            boolean flag = random.nextBoolean();
            System.out.println(flag);
        }
        for (int i = 1; i <= 10; i++) {
            double num = random.nextDouble();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            float num = random.nextFloat();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            int num = random.nextInt();
            System.out.println("第"+i+"个随机数是："+num);
        }
        for (int i = 1; i <= 10; i++) {
            long num = random.nextLong();
            System.out.println("第"+i+"个随机数是："+num);
        }
    }
}
```

- String

String的实例化

1.直接赋值的方式 String str = "Hello";

2.通过构造函数创建：String() String(String str) ...

== 用法：如果操作数是基本数据类型，==判断值是否相等

如果操作数是引用数据类型，==判断内存地址是否相等

栈内存：变量都在栈内存中

堆内存：对象都在堆内存中

Student stu = new Student();

==判断栈内存中的值是否相等

Java在堆内存中提供了一个字符串常量池，专门用来存储String类型的对象，字符串常量池有一个特点，在实例化一个String对象时，会首先在字符串常量池中查找，如果该字符串已经在常量池中创建，则直接返回该字符的内存地址，如果常量池中没有该对象，则创建然后返回内存地址。

字符串常量池中只适用于通过直接赋值方式创建的字符串对象，如果是通过构造函数创建的字符串对象，不会保存到字符串常量池中，与其他对象创建是一样的。

基于上述的原因，实际开发中比较两个字符串是否相等，不能使用==来判断，而应该使用equals方法来判断。