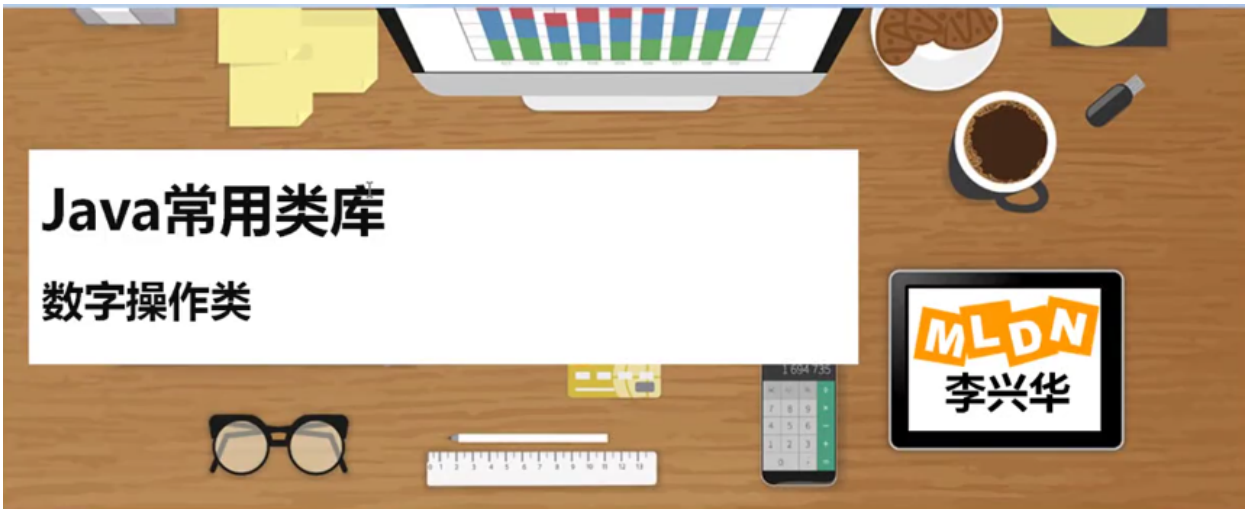


博客： https://www.cnblogs.com/HOsystem/p/14116443.html

2、具体内容

程序就是一个数学的处理过程，所以在Java语言本身也提供有相应的数字处理的类库支持。

■Math类

Math类的主要功能是进行数学计算的操作类，提供有基础的计算公式，这个类的构造方法被私有化了，但不是单例设计，而且该类之中提供的所有方法都是static型的方法，即：这些方法都可以通过类名称直接调用。

```
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		System.out.println(Math.abs(-10.1));	// 10.1
		System.out.println(Math.max(10.2, 20.3)); // 获取最大值
		System.out.println(Math.log(5));	// 1.6094379124341003
		System.out.println(Math.round(15.1));	// 15
		System.out.println(Math.round(-15.5));	// -15
		System.out.println(Math.round(-15.51));		// -16
		System.out.println(Math.pow(10.2, 20.2));	// 2.364413713591828E20
	}
}
```

虽然在Math类里面提供有四舍五入的处理方法，但是这个四舍五入在进行处理的时候是直接将小数点后的所有位进行进位处理了，这样肯定不方便，那么现在最方便的做法是可以实现指定位数的保留。

范例：自定义的四舍五入功能

```
class MathUtil {
	private MathUtil() {}
	/**
	 * 实现数据的四舍五入操作
	 * @param num 要进行四舍五入操作的数字
```

```
     * @param scale 四舍五入保留的小数位数
     * @return 四舍五入处理后的结果
     */
    public static double round(double num,int scale) {
        return Math.round(num * Math.pow(10, scale)) / Math.pow(10, scale) ;
    }
}
public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        System.out.println(MathUtil.round(19.86273, 2));
    }
}
```

Math类里面提供的基本上都是基础的数学公式，需要的时候需要自己重新整合。

■Random类

java.util.Random类的主要功能是产生随机数的，这个类主要是依靠内部提供的方法来完成：

·产生一个不大于边界的随机正整数：public int nextInt(int bound)；

范例：产生随机数

```
public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        Random rand = new Random() ;
        for (int x = 0 ; x < 10 ; x ++) {
            System.out.print(rand.nextInt(100) + "、");
        }
    }
}
```

在国内有一款神奇的所谓的36选7的彩票，那么就可以利用Random实现随机生成彩票号。

范例：随机生成彩票号

·对于这里面的数字肯定不能有0，不能够重复；

```
import java.util.Arrays;
import java.util.Random;

public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        int data [] = new int [7] ; // 开辟7个大小的空间
        Random rand = new Random() ;
        int foot = 0 ; // 操作data脚标
        while(foot < 7) {      // 选择7个数字
            int num = rand.nextInt(37) ; // 生成一个数字
            if (isUse(num,data)) {      // 该数字现在可以使用
                data[foot ++] = num ; // 保存数据
            }
        }
```

```
            java.util.Arrays.sort(data);
            for (int x = 0 ; x < data.length ; x ++) {
                System.out.print(data[x] + "、");
            }
        }
        /**
         * 判断传入的数字是否为0以及是否在数组之中存在
         * @param num 要判断的数字
         * @param temp 已经存在的数据
         * @return 如果该数字不是0并且可以使用返回true，否则返回false
         */
        public static boolean isUse(int num,int temp[]) {
            if (num == 0) {
                return false ;
            }
            for (int x = 0 ; x < temp.length ; x ++) {
                if (num == temp[x]) {
                    return false ;
                }
            }
            return true ;
        }

}
```

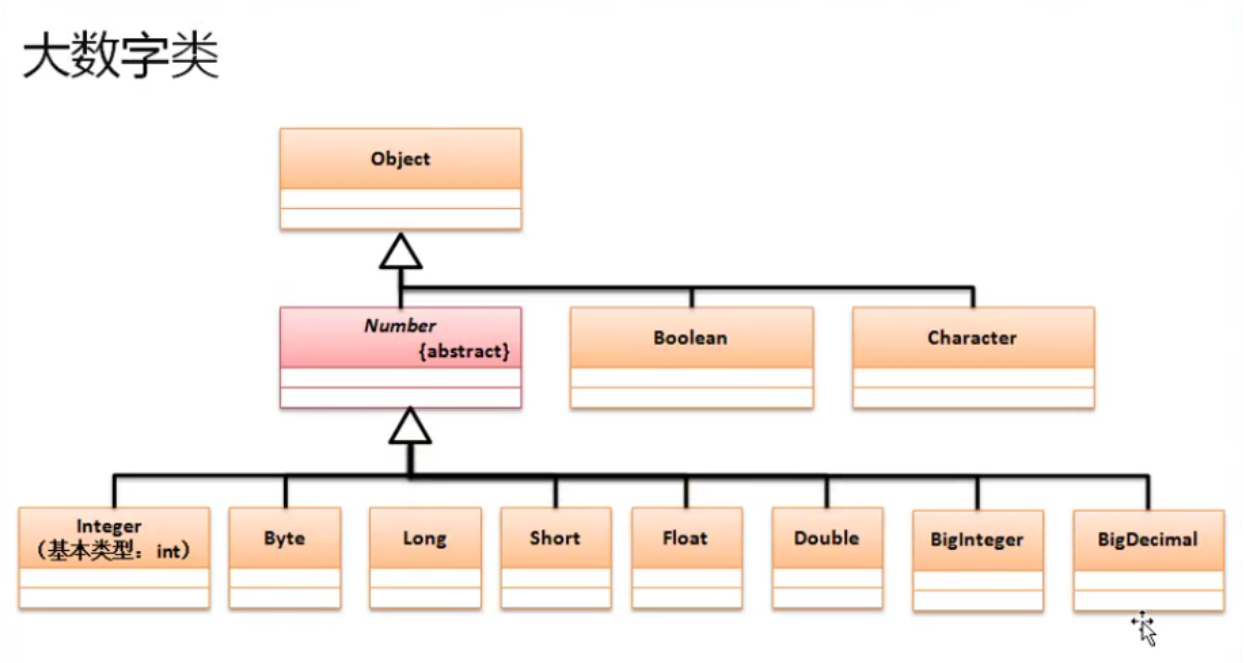
以后这种随机的操作都可以利用Random来处理。

■大数字操作数

在进行数学计算的过程里面还有一个大数字的操作类，可以实现海量数字的计算（能提供的也只是基础计算），现在假设一个数字很大，超过了double范围，那么这个时候并没有任何一种数据类型可以保存下此类的内容，最早的时候只能够通过String保存。

```
String strA = "120";
String strB = "230";
```

如果现在要想进行加法计算，那么就需要逐位拆分，每一位自己计算，而后自己独立控制进位处理，那么这样的开发难度是非常高的，所以为了解决这类问题，提供有两个大数字的操作类：BigInteger、BigDecimal。



之前分析了，当数字很大的时候只能够利用字符串描述数字操作，所以这一点可以观察两个大数字操作类的构造方法：

·BigInteger类构造：public BigInteger(String val)；

·BigDecimal类构造：public BigDecimal(String val)；

范例：使用BigInteger实现四则运算

```
import java.math.BigInteger;

public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		BigInteger bigA = new BigInteger("234234234234234234") ;
		BigInteger bigB = new BigInteger("23423423") ;
		System.out.println("加法操作：" + bigA.add(bigB));
		System.out.println("减法操作：" + bigA.subtract(bigB));
		System.out.println("乘法操作：" + bigA.multiply(bigB));
		System.out.println("除法操作：" + bigA.divide(bigB));
	}

}
```

需要注意的是，虽然提供有大数字操作类，但是整体的操作之中还是需要考虑到一个性能问题。

范例：观察性能问题

```
import java.math.BigInteger;

public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		BigInteger bigA = new BigInteger("234234234234234234") ;
		System.out.println(bigA.pow(Integer.MAX_VALUE));
	}

}
```

此时的计算过程是非常缓慢的，所以任何的电脑都是有极限的。既然在进行数学除法的时候有可能无法进行整除处理，那么就可以使用其它的除法计算来求出余数：

·求余：public BigInteger[] divideAndRemainder(BigInteger val)，数组第一个元素为商，第二个为余数；

范例：求余除法

```
import java.math.BigInteger;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		BigInteger bigA = new BigInteger("234234234234234234") ;
		BigInteger bigB = new BigInteger("23423423") ;
		BigInteger result [] = bigA.divideAndRemainder(bigB) ;
		System.out.println("商：" + result[0] + "、余数：" + result[1]);
	}
}
```

如果在开发之中真的进行计算的时候，该计算没有超过基本数据类型所包含的位数强烈不建议使用大数字类，因为这种计算性能是很差的。

BIGDecimal操作形式和BIGInteger是非常类似的，都有基础的数学支持。

范例：使用BigDecimal计算

```
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		BigDecimal bigA = new BigDecimal("32890234890") ;
		BigDecimal bigB = new BigDecimal("1892039") ;
		System.out.println("加法计算：" + bigA.add(bigB));
		BigDecimal result [] = bigA.divideAndRemainder(bigB) ;
		System.out.println("除法计算，商：" + result[0] + "、余数：" + result[1]);
	}
}
```

但是在使用BIGDecimal的时候有一个数据进位问题，在这个类里面定义有如下的除法计算：

·除法计算：public BigDecimal divide(BigDecimal divisor,int scale,RoundingMode roundingMode)；

范例：使用BigDecimal实现四舍五入处理

```
import java.math.BigDecimal;
import java.math.RoundingMode;
class MathUtil {
	private MathUtil() {}
	/**
	 * 实现数据的四舍五入操作
	 * @param num 要进行四舍五入操作的数字
	 * @param scale 四舍五入保留的小数位数
	 * @return 四舍五入处理后的结果
	 */
	public static double round(double num,int scale) {
```

```
            return new BigDecimal(num).divide(new BigDecimal(1.0), scale,
RoundingMode.HALF_UP).doubleValue();
        }
}
public class JavaAPIDemo {
        public static void main(String[] args) throws Exception {
            System.out.println(MathUtil.round(19.6352, 2));
        }
}
```

Math的处理由于使用的都是基本数据类型，所以性能一定要高于大数字处理类的。