Java Number & Math 类

一、什么是Java Number类?

二、Java Number类提供了哪些基本的数字操作?

三、什么是包装类?

四、什么是Java Math 类

Test类案例：（Math.PI 表示一个圆的周长与直径的比例，约为 3.14159~）

五、Number & Math 类方法

xxxValue用于将number对象转换为xxx数据类型的值并返回

compareTo() 方法用于将 Number 对象与方法的参数进行比较

equals()用于判断Number对象与方法的参数是否相等

valueOf() 方法用于返回给定参数的原生 Number 对象值

toString()以字符串形式返回值

parseInt()将字符串解析为int类型

abs() 返回参数的绝对值

ceil() 向上取整

floor()向下取整

round()四舍五入

rint() 返回与参数最接近的整数

min() 返回两个参数中最小值

max() 返回两个参数中最大值

random() 返回一个随机数

一、什么是Java Number类?

> Java Number类是Java中的一个抽象类，它是所有数值类型的超类，包括整数、浮点数和大数。它提供了一组用于操作数值类型的方法，如转换、比较、算术运算等。

二、Java Number类提供了哪些基本的数字操作?

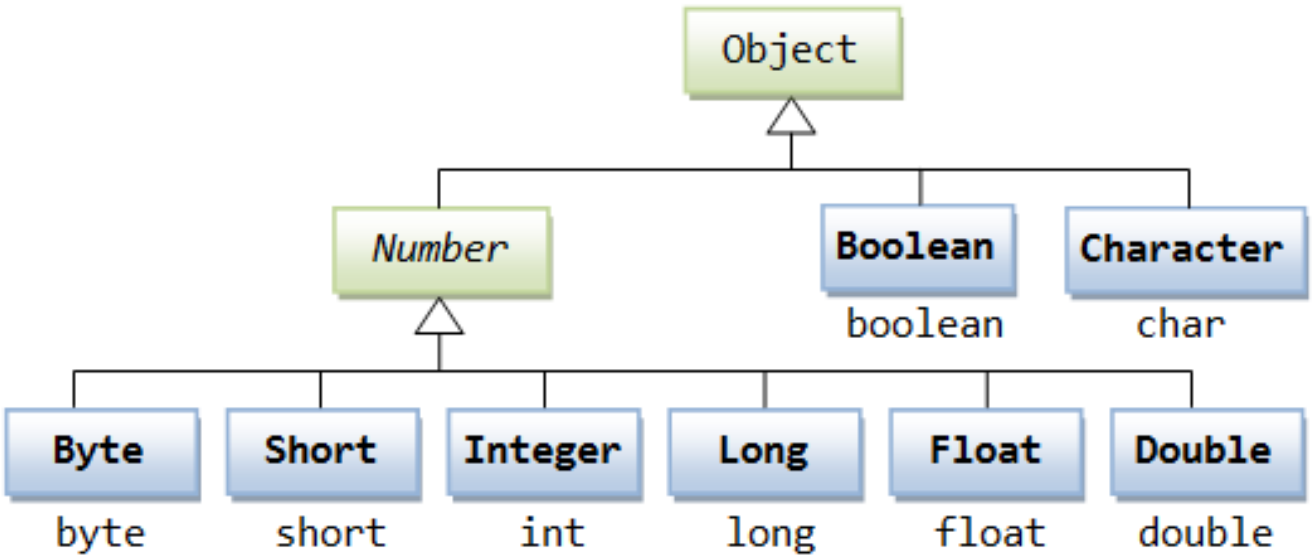
> Java Number类提供了以下基本的数字操作：转换，比较，算术运算，取整，取余，取最大值和最小值，取绝对值，取幂，取对数，取根号，取随机数，取符号，取精度，取舍入值等。

三、什么是包装类?

> 在实际开发过程中，我们经常会遇到需要使用对象，而不是内置数据类型的情形。为了解决这个问题，Java语言为每一个内置数据类型提供了对应的包装类。

> 所有的包装类都是抽象类 Number 的子类。

包装类	基本数据类型
Boolean	boolean
Byte	byte
Short	short
Integer	int
Long	long
Character	char
Float	float
Double	double



四、什么是Java Math 类

> Java 的 Math 包含了用于执行基本数学运算的属性和方法，如初等指数、对数、平方根和三角函数。
>
> Math 的方法都被定义为 static 形式，通过 Math 类可以在主函数中直接调用。

Test类案例：（Math.PI 表示一个圆的周长与直径的比例，约为 3.14159~）

Java | 复制代码

```java
public class Test {
    public static void main (String []args)
    {
        System.out.println("90 度的正弦值: " + Math.sin(Math.PI/2));
        System.out.println("0度的余弦值: " + Math.cos(0));
        System.out.println("60度的正切值: " + Math.tan(Math.PI/3));
        System.out.println("1的反正切值:  " + Math.atan(1));
        System.out.println("π/2的角度值: " + Math.toDegrees(Math.PI/2));
        System.out.println(Math.PI);
    }
}
```

五、Number & Math 类方法

xxxValue用于将number对象转换为xxx数据类型的值并返回

Java | 复制代码

```java
package com.leo.demo;

/**
 * @author Java
 */
public class MainClass {

    public static void main(String[] args) {
        Integer x = 6;
        System.out.println(x.byteValue());
        System.out.println(x.shortValue());
        System.out.println(x.intValue());
        System.out.println(x.longValue());
        System.out.println(x.floatValue());
        System.out.println(x.doubleValue());
    }
}

```

compareTo() 方法用于将 Number 对象与方法的参数进行比较

Java | 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
       // compareTo()将number对象与参数比较
        Integer x = 6;
        // 小于参数返回-1
        System.out.println(x.compareTo(5));
        // 等于参数返回0
        System.out.println(x.compareTo(6));
        // 大于参数返回1
        System.out.println(x.compareTo(8));
    }
}

```

equals()用于判断Number对象与方法的参数是否相等

Java 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
       // equals()用于判断Number对象与方法的参数是否相等
        Integer x = 6;
        System.out.println(x.equals(6.0));
        // 参数类型与值相等返回true，否则返回false
        System.out.println(x.equals(6));
        System.out.println(x.equals(8));
    }
}

```

valueOf() 方法用于返回给定参数的原生 Number 对象值

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
      // valueOf() 用于返回给定参数的原生Number对象值
      Integer i = Integer.valueOf(6);
      Long l = Long.valueOf(6);
      Double d = Double.valueOf(6);
      Float f = Float.valueOf(6);

      System.out.println(i);
      System.out.println(l);
      System.out.println(d);
      System.out.println(f);
    }
}

```

toString()以字符串形式返回值

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
      // toString()用于返回一个字符串表示的Number对象值
      Integer x = 6;
      System.out.println(x.toString());

    }
}

```

parseInt()将字符串解析为int类型

Java 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // parseInt() 将字符串解析为int类型
        System.out.println(Integer.parseInt("6"));
    }
}

```

abs() 返回参数的绝对值

Java 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // abs() 返回参数的绝对值
        System.out.println(Math.abs(-6));
    }
}

```

ceil() 向上取整

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // ceil()向上取整
        System.out.println(Math.ceil(1.3));
    }
}

```

floor()向下取整

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // floor()向下取整
        System.out.println(Math.floor(1.3));
    }
}

```

round()四舍五入

Java 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // round() 四舍五入
        System.out.println(Math.round(1.3));
        System.out.println(Math.round(1.6));
    }
}

```

rint() 返回与参数最接近的整数

Java 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // rint() 返回与参数最接近 的整数
        System.out.println(Math.rint(1.3));
        System.out.println(Math.rint(1.6));
    }
}

```

min() 返回两个参数中最小值

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // min() 返回两个参数中最小值
        System.out.println(Math.min(1, 2));
    }
}

```

max() 返回两个参数中最大值

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // max() 返回两个参数中最大值
        System.out.println(Math.max(1, 2));
    }
}

```

random() 返回一个随机数

Java | 复制代码

```java
package com.leo.demo;

/**
 * @author java
 */
public class MainClass {

    public static void main(String[] args) {
        // random() 返回一个随机数
        System.out.println(Math.random());
    }
}

```