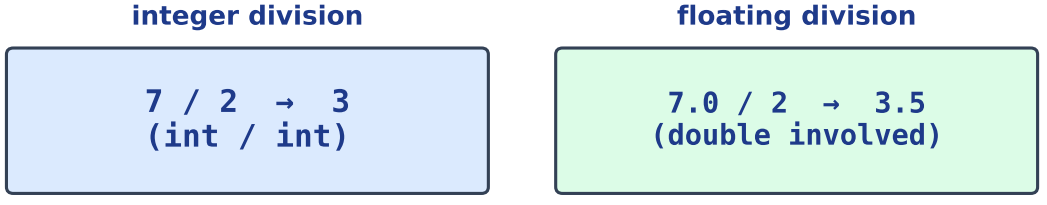
运算符与表达式

Java Reference

算术与赋值

Java 的算术 (arithmetic) 用 `+ - * / ` 和 `%`(取余)。两个 `int` 相除时,`/` 是整数除法 (integer division)— 会丢掉小数。`+=`、`-=`、`++` 是简写。

```
public class Main {
    public static void main(String[] args) {
        int a = 7, b = 2;
        System.out.println(a / b);    // 3 (integer division)
        System.out.println(a % b);    // 1
        double x = 7.0 / 2;           // 3.5 (one side is double)
        System.out.println(x);
    }
}
```



Exam tip: int/int drops the decimal; make one side double for a real quotient; % is remainder

int/int 会截断；涉及 *double* 以获得实数商

使用对象:String、Math、包装类

有些值是带方法的对象 (object)。`String` 有 `.length()`、`.substring()`、`.toUpperCase()`。`Math` 有 `Math.max`、`Math.sqrt`、`Math.pow`。包装类 (wrapper class)(`Integer`、`Double`) 把基本类型包起来 — 例如 `Integer.parseInt("42")`。

```
public class Main {
    public static void main(String[] args) {
        String s = "Hello";
        System.out.println(s.length());                  // 5
        System.out.println(s.toUpperCase());             // HELLO
        System.out.println(Math.max(3, 9));              // 9
        System.out.println(Integer.parseInt("42") + 1);  // 43
    }
}
```

`Math.random()` 返回一个随机整数 `double`，范围为 `0.0` 到（但不包括）`1.0`。对其进行缩放并强制类型转换以获得整数 — 这是 AP 编程的习惯用法:

```
public class Main {
    public static void main(String[] args) {
        // a random whole number from 1 to 6 (a dice roll)
        int roll = (int) (Math.random() * 6) + 1;
        System.out.println(roll >= 1 && roll <= 6);   // true
    }
}
```

- `Integer.MAX_VALUE`(2147483647) 和 `Integer.MIN_VALUE` 是 `int` 的上下限; 超过就会绕回 (溢出)。
- 一个还没有指向任何对象的对象变量是 `null`; 对它调用方法会抛出 NullPointerException。

强制转换与类型转换

强制转换 (cast) 改变一个值的类型。`(int)` 丢掉小数; 当你需要精确结果时,`(double)` 可以避免整数除法。

```
public class Main {
    public static void main(String[] args) {
        double pi = 3.99;
        System.out.println((int) pi);            // 3
        int total = 7, n = 2;
        System.out.println((double) total / n);  // 3.5
    }
}
```

常见错误

- 整数除法:`5 / 2` 得 2, 不是 2.5。先转换:`(double) 5 / 2`。
- 比较字符串 (和其他对象) 用 `.equals()`, 不是 `==`。
- 对两个对象用 `==` 判断的是它们是不是**同一个**对象, 而不是看起来是否相等。