类型转换与类型提升

▼ **整数除法是地板除**

✅ 当两个整数相除时，**小数部分会被截断**，保留整数部分 👉 **向下取整 地板除**

```
System.out.println(7 / 2);  // 输出 3
```

- `7 / 2` 都是 `int`，所以执行的是整数除法
- 小数部分 `.5` 被截断了

✅ 如果想得到小数，至少要有一个是 `float` 或 `double`

```
System.out.println(7.0 / 2);  // 输出 3.5
```

▼ **自动类型提升**

✅ 在 Java 中进行表达式运算时，Java 会将参与运算的不同类型**自动转换成精度最高的那个类型**，再进行计算 👉 这叫做“自动类型提升”

💡 Java 中的类型精度等级：

```
byte < short < int < long < float < double
```

```
int a = 5;
double b = 2.0;
System.out.println(a + b);  // 输出 7.0
```

- `a` 会被提升为 `double`，变成 `5.0`
- 然后再进行 `double + double = double`

▼ **丢失精度时需要进行强制转换**

当你将一个**高精度类型赋值给低精度类型时**，Java 不会自动完成，必须你手动 **强制类型转换**，以防止不小心丢失精度

```
double pi = 3.14159;
int x = (int) pi;  // 需要强制转换
System.out.println(x);  // 输出 3
```

- `double` 精度高于 `int`
- `强制转换 (int) pi` 会截断小数部分，变成 3

⚠️ 注意：**强转可能导致数据丢失、溢出、精度问题**

▼ **char 参与计算时使用 ASCII 码（Unicode码）**

- 在 Java 中，`char` 是一个 2 字节的整数类型（本质上是 `int` 范畴）
- 它的值其实就是对应字符的 **Unicode 编码值**
- 所以 `char` 参与运算时，会**自动当作数字参与运算**

```
char c = 'A';  // Unicode 值为 65
int x = c + 1;
System.out.println(x);     // 输出 66
System.out.println((char)x); // 输出 'B'
```

```
System.out.println('a' + 1);      // 输出 98
System.out.println((char)('a' + 1)); // 输出 'b'
```

▼ Java 中自动类型提升常见触发点

情况	是否自动提升
`int + double` → `double`	✅ 自动
`byte + byte` → `int`	✅ 自动
`float → int`	❌ 需强转 `(int)`
`char + int` → `int`	✅ 自动