1. The **Integer Division** Trap

Imagine you want to calculate the average of 5 and 2. You might write:

```
double average = 5 / 2;
System.out.println(average);
```

You might expect `2.5`, right? But Java prints:

```
2.0
```

Why?

Because `5` and `2` are both **integers** (`int`). When Java divides two integers, it **truncates** the decimal part—no rounding, no guessing—just chops it off. So:

```
5 / 2 → 2
```

Then it stores that integer `2` into a `double` variable, turning it into `2.0`.

Solution: Use **Type Casting** to tell Java to treat one number as a decimal.

---

2. Implicit Casting (The "Widening" Safe Zone)

Think of **data types** as cups:

- `int` → medium cup
- `double` → large cup

Pouring a medium cup of water into a large cup is safe: nothing spills.

Java does this automatically when converting a smaller type to a larger type:

```
int myInt = 9;
// Automatically converted from int → double
double myDouble = myInt;

System.out.println(myDouble); // 9.0
```

☑ No data is lost. This is called **implicit casting**.

---

3. Explicit Casting (The "Narrowing" Danger Zone)

What if you try to pour a **large cup** (`double`) into a **medium cup** (`int`)?

Some water (the decimal part) will spill. Java won't do this automatically because it can lose data. You have to **force it** using **explicit casting**:

```
double exactPrice = 9.99;

// Force double → int
int roundedPrice = (int) exactPrice;

System.out.println(roundedPrice); // 9
```

Notice that `.99` is completely gone.

⚠ This is called **explicit casting**, and it's your responsibility to be careful.

---

4. Fixing the Division Trap

We can combine casting with division to get the correct average:

```
double average = (double) 5 / 2;
System.out.println(average); // 2.5
```

By casting **one number to double** *before dividing*, Java performs decimal division instead of integer division.

💡 **Tip:** You only need to cast **one operand**, not both.

☑ Quick Summary

Concept	Example	Output	Notes
Integer Division Trap	`5 / 2`	2	Truncates decimals
Implicit Casting	`int → double`	9.0	Safe widening
Explicit Casting	`double → int`	9	Narrowing, may lose data
Fix Division Trap	`(double)5 / 2`	2.5	Cast one operand to perform decimal division

---

 Java Casting Diagram: Implicit vs Explicit

Related Topics

- Java Expressions
- Conditional Statements