

Operators & expressions

Java Reference

Arithmetic & assignment

Java arithmetic uses + - * / and % (remainder). With two ints, / is integer division — it drops the decimal. +=, -=, and ++ are shortcuts.

```
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);
    }
}
```

integer division

7 / 2 → 3
(int / int)

floating division

7.0 / 2 → 3.5
(double involved)

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

int/int truncates; involve a double for a real quotient

Using objects: String, Math, wrappers

Some values are objects with methods. String has .length(), .substring(), .toUpperCase(). Math has Math.max, Math.sqrt, Math.pow. Wrapper classes (Integer, Double) wrap a primitive — e.g. 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() returns a random double from 0.0 up to (but not including) 1.0. Scale it and cast to get whole numbers — this is the AP idiom:

```
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) and Integer.MIN_VALUE are the limits of int; going past them wraps around (an overflow).
- An object variable that points at no object holds null; calling a method on it throws a NullPointerException.

Casting & type conversion

A cast changes a value's type. `(int)` drops the decimal; `(double)` avoids integer division when you need an exact result.

```
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  
    }  
}
```

Common mistakes

- Integer division: `5 / 2` is 2, not 2.5. Cast first: `(double) 5 / 2`.
- Compare Strings (and other objects) with `.equals()`, not `==`.
- `==` on two objects tests whether they are the SAME object, not whether they look equal.