Loops

COP2250: Java Programming

Kevin Pyatt, Ph.D.

State College of Florida
Pyatt Labs

Week 6

Today's Objectives

- Understand the three loop types: `while`, `do-while`, `for`
- Recognize when to use each loop
- Use sentinel values to control input
- Accumulate totals and counts inside loops
- Avoid infinite loops and off-by-one errors

The Problem: Repetition Without Loops

Print "Hello" 5 times without a loop:

```
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
System.out.println("Hello");
```

Now do it 1,000 times?
Loops let you repeat code without copying it.

The `while` Loop

Syntax:

```
while (condition) {
    // body executes while condition is true
}
```

Example — count from 1 to 5:

```
int count = 1;
while (count <= 5) {
    System.out.println(count);
    count++;
}
```

- Checks condition **before** each iteration
- Body may execute **zero** times
- Must update the loop variable or you get an infinite loop

Sentinel Values

A **sentinel value** signals "stop reading input."

```
System.out.print("Enter a number (0 to quit): ");
int num = input.nextInt();

while (num != 0) {
    // process num
    System.out.print("Enter a number (0 to quit): ");
    num = input.nextInt();
}
```

- The sentinel (0) is **not** processed as data
- Read **before** the loop, then again at the **end** of the loop body
- This is called a **priming read**

The do-while Loop

Syntax:

```
do {
    // body executes at least once
} while (condition);
```

Example — input validation:

```
int choice;
do {
    System.out.print("Enter 1, 2, or 3: ");
    choice = input.nextInt();
} while (choice < 1 || choice > 3);
```

- Checks condition **after** each iteration
- Body always executes **at least once**
- Great for menus and input validation

The for Loop

Syntax:

```
for (init; condition; update) {
    // body
}
```

Example — count from 1 to 5:

```
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}
```

All three parts in one line:

- **Init**: `int i = 1` — runs once
- **Condition**: `i <= 5` — checked before each iteration
- **Update**: `i++` — runs after each iteration

Which Loop Should I Use?

Scenario	Loop	Why
Known count	`for`	Count is in the syntax
Unknown count, may skip	`while`	Check before running
Must run at least once	`do-while`	Check after running
Sentinel-controlled	`while`	Priming read pattern
Input validation	`do-while`	Must prompt at least once

Rule of thumb: If you know how many times — `for`.
If you don't — `while` or `do-while`.

Accumulators and Counters

Counter: tracks how many times something happens.
Accumulator: tracks a running total.

```
int count = 0;
double total = 0;

// inside the loop:
count++;           // counter
total += value;    // accumulator
```

Average pattern:

```
double average = total / count;
```

- Initialize **before** the loop
- Update **inside** the loop
- Use **after** the loop

Common Mistakes

1. **Infinite loop** — forgetting to update:

```
int i = 1;
while (i <= 5) {
    System.out.println(i);
    // missing i++
}
```

2. **Off-by-one error:**

```
// Prints 0-4, not 1-5
for (int i = 0; i < 5; i++) { ... }
```

3. **Semicolon after `for`:**

```
for (int i = 0; i < 5; i++); // empty loop!
{
    System.out.println(i); // runs once
}
```

Today's Lab

Number Analyzer

- Read integers from the user until sentinel (0)
- Count positive and negative numbers separately
- Compute the running total
- Calculate and display the average (excluding zeros)
- Handle edge case: no numbers entered

Concepts used: `while` loop, sentinel value, accumulators, counters, conditional logic inside a loop.

Demonstrate working code before leaving!