Name: ______________________

Java For Loops

What are For Loops?

A **for loop** repeats a block of code a specific number of times. It's perfect when you know exactly how many times you want to repeat something. For loops are more organized than while loops because they keep the counter, condition, and update all in one place.

Simple Analogy: For Loops are Like Doing Push-ups

Imagine you want to do exactly 10 push-ups:

- **Start:** "I'll start counting at 1" (initialization)
- **Condition:** "I'll keep going while my count is 10 or less" (condition)
- **Action:** "Do one push-up" (loop body)
- **Count:** "Add 1 to my count" (increment)

You do: push-up 1, count up → push-up 2, count up → ... → push-up 10, count up → stop (because 11 > 10)

For Loop Process:
Set counter → Check condition → Do action → Update counter → Check condition again → Repeat until done

Basic Pattern

```
for (initialization; condition; update) { // action to repeat }
```

Three parts in the parentheses:

- **Initialization:** Set up your counter variable
- **Condition:** Keep going while this is true
- **Update:** Change the counter after each loop

Example 1: Counting from 1 to 5

```
for (int i = 1; i <= 5; i++) { System.out.println("Count: " + i); } System.out.println("Done
counting!");
```

Breaking it down:

- `int i = 1` - Start with i = 1
- `i <= 5` - Keep going while i is 5 or less
- `i++` - Add 1 to i after each loop

Output: Count: 1, Count: 2, Count: 3, Count: 4, Count: 5, Done counting!

Example 2: Counting backwards

```
for (int countdown = 10; countdown >= 1; countdown--) { System.out.println(countdown); }
```

```
System.out.println("Blast off!");
```

What's different:

- Start at 10 instead of 1
- Use >= instead of <=
- Use `countdown--` to subtract 1 instead of add 1

Example 3: Skipping numbers (counting by 2s)

```
for (int i = 0; i <= 10; i += 2) { System.out.println("Even number: " + i); }
```

Output: Even number: 0, Even number: 2, Even number: 4, Even number: 6, Even number: 8, Even number: 10

Note: `i += 2` means "add 2 to i" (same as `i = i + 2`)

For Loop vs While Loop: Same Result, Different Style

For Loop Style:

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

While Loop Style:

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

Both do the same thing! For loops just keep everything organized in one line.

Advanced For Loop: Enhanced For Loop (for-each)

When you want to go through every item in an array or list, you can use a simpler for loop:

Regular For Loop with Arrays:

```
int[] numbers = {10, 20, 30, 40, 50}; for (int i = 0; i < numbers.length; i++) {
System.out.println("Number: " + numbers[i]); }
```

Enhanced For Loop (easier way):

```
int[] numbers = {10, 20, 30, 40, 50}; for (int num : numbers) { System.out.println("Number: "
+ num); }
```

Read as: "For each number (num) in the numbers array, do this..."

Common For Loop Patterns

Pattern	Code	What It Does

Count up	for (int i = 0; i < 10; i++)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Count down	for (int i = 10; i > 0; i--)	10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Skip by 2	for (int i = 0; i < 10; i += 2)	0, 2, 4, 6, 8
Through array	for (int i = 0; i < arr.length; i++)	Goes through every array position
Enhanced (for-each)	for (int item : array)	Gets every item directly

Real-Life Example: Calculating Total Cost

```
double[] prices = {2.99, 1.50, 4.25, 3.75}; double total = 0; for (double price : prices) {
total += price; System.out.println("Added $" + price + ", total now: $" + total); }
System.out.println("Final total: $" + total);
```

Nested For Loops: Making a Multiplication Table

```
for (int row = 1; row <= 3; row++) { for (int col = 1; col <= 3; col++) {
System.out.print(row + " x " + col + " = " + (row * col) + " "); } System.out.println(); //
New line after each row }
```

Output:
```
1 x 1 = 1 1 x 2 = 2 1 x 3 = 3
2 x 1 = 2 2 x 2 = 4 2 x 3 = 6
3 x 1 = 3 3 x 2 = 6 3 x 3 = 9
```

When to Use For Loops vs While Loops

Use For Loops When:	Use While Loops When:
You know how many times to repeat	You don't know how many times to repeat
You're counting or going through items	You're waiting for something to change
Working with arrays or lists	Getting user input until they say "quit"
Making patterns or tables	Reading files until the end

Important Rules to Remember

- The three parts of a for loop are separated by **semicolons** (;)
- You can leave any of the three parts empty, but you still need the semicolons
- The variable you create in the for loop (like `int i`) only exists inside the loop
- Array indexes start at 0, so use `i < array.length` not `i <= array.length`
- Enhanced for loops are easier but you can't modify the original array
- Nested loops multiply: 3 outer loops × 4 inner loops = 12 total runs

Your Turn: Write Your Own Definition

What is a for loop? How would you explain it to a friend?

Write your definition in your own words:

Think of a real-life situation where you do something a specific number of times. Describe it:

Example: "I brush my teeth for exactly 2 minutes, counting 'one Mississippi, two Mississippi...' up to 120."

Your example:

When would you choose a for loop instead of a while loop?