Chapter 4

Exercise 1a

i = 1;

while (i <= 10) {

++i;

}

There's a closing brace } at the end, but there's **no matching opening brace** {.

This causes a **syntax error**.

b

for (int i = 1; i <= 10; i++)

System.out.println(i \* 0.1);

|  |  |
| --- | --- |
| Comparing floats with != | Use int loop and scale instead |

|  |  |
| --- | --- |
| Floating-point imprecision | Avoid comparing floats directly |

c

switch (n) {

case 1:

System.out.println("The number is 1");

break;

case 2:

System.out.println("The number is 2");

break;

default:

System.out.println("The number is not 1 or 2");

break;

}

|  |  |
| --- | --- |
| Missing break in case 1 | Add break; after case 1 block |

|  |  |
| --- | --- |
| Unintended fall-through | Use break to prevent it |

d

n = 1;

while (n <= 10)

System.out.println(n++);

This ensures the loop includes 10 in the output.

Exercise 2

4.5

There are four basic elements of counter-controlled repetitioin:

**1. Initialization of the Counter**

* Set the counter variable to a **starting value** before the loop begins.
* Example: int i = 1;

**2. Loop-Continuation Condition**

* A **condition** that is tested before each iteration to determine whether the loop should continue.
* The loop executes as long as this condition is **true**.
* Example: i <= 10

**3. Update (Increment or Decrement) of the Counter**

* The counter is **modified** during each loop iteration, usually with ++ or --.
* This helps move toward the loop termination.
* Example: i++

**4. Loop Body (Statements to Execute)**

* The **set of actions or code** that runs on each iteration of the loop.
* This is the main task you want to repeat.
* Example: System.out.println(i);

4.6

**Comparison: while vs for Loops**

| **Feature** | **while Loop** | **for Loop** |
| --- | --- | --- |
| **Purpose** | Best when the number of iterations is **unknown** | Best when the number of iterations is **known** |
| **Structure** | More flexible, but requires manual setup | Compact structure with all control parts in one line |
| **Syntax** | while (condition) { ... } | for (init; condition; update) { ... } |
| **Initialization** | Done **before** the loop separately | Done **within** the loop header |
| **Condition Check** | At the top of the loop | Also at the top |
| **Update (increment)** | Must be done **inside** the loop manually | Done in the loop header |
| **Readability** | More readable when loop logic is complex | More readable when doing simple counting |
| **Use Case Example** | Waiting for user input, file reading | Looping from 1 to 10, iterating over arrays |

Use **for** loops when you know exactly **how many times** to repeat.

Use **while** loops when the number of repetitions is **not known ahead of time** or depends on a condition that changes during execution.

4.7

**Use a do…while statement when you want the loop to execute at least once — no matter what**

The key difference is:

* **while** checks the condition **before** running the loop.
* **do...while** checks the condition **after** running the loop — so the loop **always runs at least once**.

| **Use do...while when:** | **You need the loop to run at least once regardless of the condition.** |
| --- | --- |

|  |  |
| --- | --- |
| Use while when: | The condition **might be false from the beginning**, and skipping the loop is okay. |

4.8

**break vs continue — Comparison Table**

| **Feature** | **break** | **continue** |
| --- | --- | --- |
| **What it does** | **Exits** the loop entirely | **Skips the rest** of the current iteration and continues with the next one |
| **Affects** | The **whole loop** | Only the **current iteration** |
| **Used in** | for, while, do...while, and switch | for, while, do...while |
| **Typical use case** | Exit early when a condition is met (e.g., found item) | Skip over certain values (e.g., ignore odd numbers) |
| **Can be confusing if overused?** | ✅ Yes – can harm readability if not used clearly | ✅ Yes – especially if nested or poorly documented |

4.9 a

for (int i = 100; i >= 1; i--) {

System.out.println(i);

}

b

switch (value % 2) {

case 0:

System.out.println("Even integer");

break;

case 1:

System.out.println("Odd integer");

break;

}

c

for (int i = 19; i >= 1; i -= 2) {

System.out.println(i);

}

d

int counter = 2;

do {

System.out.println(counter);

counter += 2;

} while (counter <= 100);

4.10

**Step-by-Step Explanation:**

1. **Outer for loop** (for (int i = 1; i <= 10; i++)):
   * This loop runs 10 times, where i goes from **1** to **10**.
   * In each iteration, it will execute the inner loop and then print a newline (System.out.println();).
2. **Inner for loop** (for (int j = 1; j <= 5; j++)):
   * This loop runs 5 times, where j goes from **1** to **5**.
   * In each iteration of the inner loop, it prints the **@** symbol without moving to the next line (System.out.print('@');).
3. **New line (System.out.println();)**:
   * After the inner loop finishes printing 5 @ symbols on the same line, the program prints a **newline** to move to the next line before starting the next iteration of the outer loop.

**What the Program Does:**

* The program prints **10 lines**.
* Each line contains **5 @ symbols**.

4.32

To solve this, we'll use the **compound growth formula**, similar to the one used in interest calculations:

**Compound Growth Formula:**

A=P×(1+r)nA = P \times (1 + r)^nA=P×(1+r)n

Where:

* AAA = Future value (target user base)
* PPP = Initial value (starting user base)
* rrr = Growth rate per month (as a decimal)
* nnn = Number of months (what we're solving for)

We’ll solve this **logarithmically** for nnn:

n=log⁡(A/P)log⁡(1+r)n = \frac{\log(A / P)}{\log(1 + r)}n=log(1+r)log(A/P)​

**Given:**

* Initial user base P=1,000,000,000P = 1,000,000,000P=1,000,000,000 (1 billion)
* Growth rate r=4%=0.04r = 4\% = 0.04r=4%=0.04
* Target 1: A=1,500,000,000A = 1,500,000,000A=1,500,000,000
* Target 2: A=2,000,000,000A = 2,000,000,000A=2,000,000,000

**Step-by-step Calculations (in Java style or calculator):**

**🔸 To reach 1.5 billion:**

n=log⁡(1.5)log⁡(1.04)≈0.17610.0170≈10.36 monthsn = \frac{\log(1.5)}{\log(1.04)} ≈ \frac{0.1761}{0.0170} ≈ 10.36 \text{ months}n=log(1.04)log(1.5)​≈0.01700.1761​≈10.36 months

**🔸 To reach 2 billion:**

n=log⁡(2)log⁡(1.04)≈0.30100.0170≈17.70 monthsn = \frac{\log(2)}{\log(1.04)} ≈ \frac{0.3010}{0.0170} ≈ 17.70 \text{ months}n=log(1.04)log(2)​≈0.01700.3010​≈17.70 months

**Final Answers:**

* **Facebook will reach 1.5 billion users in approximately 11 months**
* **Facebook will reach 2 billion users in approximately 18 months**