**Chapter 4**

Exercise 1

1. Error:

* The semicolon ; after while (i <= 10) terminates the loop immediately.
* Also, the closing brace } has no matching opening brace.

Fixed code:

i = 1;

while (i <= 10);

++i;

}

1. Error:

* Using k != 1.0 with floating-point numbers (double) is unreliable due to precision issues.
* Variable k is undeclared.

Fixed code:

for (k = 0.1; k != 1.0; k += 0.1)

System.out.println(k);

1. Error:

* Missing break after case 1, so it “falls through” and executes case 2 and default.

Fixed code:

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;

}

1. Error:

**4.5 – The Four Basic Elements of Counter-Controlled Repetition**

1. **Initialization**: A control variable is set to a starting value before the loop begins.
2. **Condition (Loop Continuation Test)**: A boolean expression is tested before each iteration; the loop continues while the condition is true.
3. **Increment/Decrement**: The control variable is modified (usually incremented or decremented) at each iteration to eventually meet the termination condition.
4. **Body of the Loop**: The set of statements that are executed repeatedly as long as the condition is true.

**4.6 – Compare and Contrast the while and for Repetition Statements**

**Similarities**:  
Both the while and for loops are used to create repetition structures in code, allowing a block of statements to be executed multiple times based on a condition.

**Differences**:

* **while loop** runs as long as a condition remains true. It is ideal when the number of iterations is **unknown or dynamic** before the loop starts.
* **for loop** is typically used when the number of iterations is **known in advance**. It combines initialization, condition-checking, and increment/decrement in a single line, making it more concise for count-controlled loops.

In summary, use while when the loop should continue based on a condition that may change during execution, and use for when you can determine the loop count beforehand.

**4.7 – Appropriate Use of** do…while **Statement**

The do…while loop guarantees that the loop body will execute **at least once** before the condition is checked. This makes it ideal for situations where you want the code to run first, then check if it should continue.

A common example is prompting a user for input. You may want to ask the user **at least once**, and then continue asking **only if the input is invalid**. In such a case, using a do…while loop ensures the prompt shows up before the condition is checked.

In contrast, a regular while loop checks the condition first, which might prevent the loop body from running even once if the condition is false from the beginning.

### ****Scenario – User Menu Selection****

Imagine you're building a simple console application that displays a menu of options to the user and performs an action based on their choice. You want the menu to be displayed **at least once**, and keep showing again **until the user selects "Exit"**.

4.8 - **Break vs. Continue – Comparison**

* **break statement**  
  Ends the loop or switch statement **entirely** as soon as the specified condition is met. No further iterations or cases are executed.
* **continue statement**  
  **Skips the current iteration** of the loop when the condition is met, and jumps straight to the **next cycle** of the loop (re-checking the condition in while/for loops).