Chapter 4:

Exercise 1

A: **Error:** The semicolon after the while condition is an error. This creates an empty loop, and the loop body (++i;) is executed only once after the loop finishes. Also, there's an extra closing curly brace }.

int i = 1; // You should also declare the type of 'i'

while (i <= 10) {

i++; // or i = i + 1;

}

B: **Error:** Comparing floating-point numbers for equality (!=) can be unreliable due to potential rounding errors. This loop might not terminate as expected.

for (double k = 0.1; k < 1.0; k += 0.1) { // Change != to <

System.out.println(k);

}

C: **Error:** Missing break statements in the case 1 clause. This causes the code to "fall through" to the next case.

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;

}

D: **Error:** The code should print values from 1 to 10, but the loop condition n < 10 will only print up to 9.

int n = 1; //Declare the type of n

while (n <= 10) { // Change < to <=

System.out.println(n++);

}

Exercise 2:

4.5: **our Basic Elements of Counter-Controlled Repetition**

The four basic elements of counter-controlled repetition are:

1. **Initialization:** Sets the initial value of the control variable.
2. **Condition:** The loop-continuation condition that determines whether to continue looping.
3. **Increment/Decrement:** Modifies the control variable after each iteration.
4. **Loop Body:** The statements that are executed repeatedly.

4.6: **while vs. for Loops**

* **while loop:** A while loop is used when the number of iterations is not known in advance. It continues to execute as long as the condition is true.
* **for loop:** A for loop is typically used when the number of iterations is known or can be easily calculated. It provides a concise way to initialize, test, and modify a control variable.

4.7: **do...while vs. while**

The do...while loop is more appropriate when you need to execute the loop body at least once. It tests the loop-continuation condition *after* executing the loop body.

4.8: **break vs. continue**

* **break statement:** Immediately exits the loop, skipping the remaining statements in the loop body and the loop-continuation condition.
* **continue statement:** Skips the rest of the current iteration and proceeds to the next iteration of the loop

4.9:

**A:**

**Error:** The comma (,) is not the correct separator for the for loop initialization and update. It should be a semicolon (;)

for (int i = 100; i >= 1; i--) //Also you should declare the type of i and decrement i

System.out.println(i);

B:

 **Error:** Missing break statements in the case clauses, causing fall-through

switch (value % 2) {

case 0:

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

break;

case 1:

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

break;

}

C:

**Error:** The loop increment is incorrect. To get odd numbers from 19 to 1, you should decrement by 2. Also, i needs to be declared.

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

System.out.println(i);

D:

**Error:** The while keyword is case-sensitive and should be lowercase. Also, counter needs to be declared.

int counter = 2;

do {

System.out.println(counter);

counter += 2;

} while (counter <= 100); //To include 100

4.10:

This program prints a pattern of @ symbols. The outer loop iterates 10 times, and the inner loop prints five @ symbols without a newline. The System.out.println(); after the inner loop moves the cursor to the next line. So, it prints 10 rows of five @ symbols each.