Chapter 3:

3.10. Similarities**:** Both if and while statements control the flow of execution based on a condition. They both

use a boolean expression to determine whether a block of code should be executed.

Differences**:**

* The if statement executes a block of code only once if the condition is true.
* The while statement repeatedly executes a block of code as long as the condition is true.

3.11. When dividing one integer by another in Java, if the result is not a whole number, the fractional part of the result is truncated (discarded).

Howtoavoidtruncation**:** To avoid this, you can use floating-point numbers (like float or double) in the division. If at least one of the operands is a floating-point number, the result will be a floating-point number.

3.12. Control statements can be combined by:

* Nesting**:** Placing one control statement within another.
* Sequencing**:** Placing control statements one after the other.

3.13. A for loop would be appropriate because the number of repetitions is known (100).

**What type would be appropriate for calculating the sum of an arbitrary number of positive integers?**

* A while loop would be appropriate because the number of repetitions is not known in advance. The loop would continue until a specific condition is met (e.g., the user enters a sentinel value).

**Briefly describe how each of these tasks could be performed.**

* **Sum of first 100 positive integers:**
  + Initialize a variable sum to 0.
  + Use a for loop that iterates from 1 to 100.
  + In each iteration, add the loop counter to sum.
  + After the loop, sum will contain the sum of the first 100 positive integers.
* **Sum of an arbitrary number of positive integers:**
  + Initialize a variable sum to 0.
  + Use a while loop that continues until the user enters a sentinel value (e.g., -1) to indicate the end of input.
  + In each iteration, read an integer from the user.
  + If the integer is not the sentinel value, add it to sum.
  + After the loop, sum will contain the sum of the entered integers

3.14. Preincrementing**:** The value of the variable is incremented *before* it is used in an expression. Postincrementing**:** The value of the variable is incremented *after* it is used in an expression.

3.15. Error**:** The semicolon after the if condition terminates the if statement prematurely. This means the System.out.println statement will always execute. Also, there is a parenthesis error in the else statement

Correction**:**

Java

if (age >= 65) {

System.out.println("Age is greater than or equal to 65");

} else {

System.out.println("Age is less than 65");

}

3.15 b)

Error**:** total is not initialized before being used in the while loop.

Correction**:**

Java

int x = 1, total = 0; // Initialize total

while (x <= 10) {

total += x;

++x;

}

3.15 c) Error**:** The loop body is not enclosed in curly braces, so only the first statement (total += x) is part of the loop. Also, x and total are not initialized.

Correction**:**

Java

int x = 1; // Initialize x

int total = 0; // Initialize total

while (x <= 100) {

total += x;

++x;

}

3.15 d).

Error**:** If y starts with a positive number, the condition y > 0 will always be true, resulting in an infinite loop.

Correction**:**

Java

int y = 10; // Initialize y to a positive number

while (y > 0) {

System.out.println(y);

--y; // Decrement y

}

Exercise 2

Check the Java code