CHAPTER 3 ANSWERS

**Exercise 3.10**:

**Comparison of if and while statements**

* **Similarity**:  
  Both if and while statements use a **Boolean condition** to control the execution of code. If the condition evaluates to true, the code inside the block executes.
* **Differences**:
  + if is a **single-selection statement**: It checks the condition once and executes the corresponding block if the condition is true. Otherwise, it skips the block.
  + while is a **repetition statement**: It checks the condition and keeps executing the block **repeatedly** as long as the condition remains true. If the condition is false at the start, the block may not execute at all.

**Exercise 3.11**:

When dividing one integer by another in Java:

The result is **also an integer** because Java performs **integer division** when both operands are integers. The **fractional part is discarded** (not rounded, just truncated.). To retain the fractional part, at least one operand must be a **floating-point number (double or float)**. This can be done using **explicit type casting** or declaring at least one operand as a double.

**Exercise 3.12:**

There are **two main ways** to combine control statements:

#### **Sequential Execution**

* Statements execute **one after another** in the order they appear.
* Example:

java

CopyEdit

System.out.println("Start");

int a = 5; // First statement

int b = 10; // Second statement

System.out.println(a + b); // Third statement

#### **Nesting (Control Statements Inside Other Control Statements)**

* You can nest if, while, for, and other statements inside each other.

##### **Example: Nested** if **inside a** while

java

CopyEdit

int x = 1;

while (x <= 5) { // Outer loop

if (x % 2 == 0) { // Nested if statement

System.out.println(x + " is even");

}

x++;

}

**Exercise 3.13**.

### ****Choosing the Right Repetition Statement****

We have two different cases to consider:

#### **1. Calculating the Sum of the First 100 Positive Integers**

* Since we know exactly how many numbers (100), a **for loop** is best.
* Example:

java

CopyEdit

int sum = 0;

for (int i = 1; i <= 100; i++) {

sum += i;

}

System.out.println("Sum of first 100 positive integers: " + sum);

* This runs 100 times and adds each number to sum.

#### **2. Calculating the Sum of an Arbitrary Number of Positive Integers**

* Since we **don’t know** how many numbers the user will enter, we use a **while loop with a sentinel value** (e.g., user enters -1 to stop).
* Example:

java

CopyEdit

import java.util.Scanner;

public class SumCalculator {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int sum = 0, number;

System.out.println("Enter a positive number (or -1 to stop):");

number = input.nextInt();

while (number != -1) { // Loop until user enters -1

sum += number;

System.out.println("Enter another number (or -1 to stop):");

number = input.nextInt();

}

System.out.println("Total sum: " + sum);

input.close();

}

}

* This keeps asking for numbers until the user enters -1 to stop

### ****Exercise 3.14: Preincrement vs. Postincrement in Java****

In Java, there are two ways to increment a variable:

#### **1. Preincrement (**++variable**)**

* The variable is **incremented first**, then its updated value is used in the expression.

#### **2. Postincrement (**variable++**)**

* The **original value** is used in the expression **first**, then the variable is incremented.

### ****Example to Demonstrate the Difference****

java

CopyEdit

public class IncrementExample {

public static void main(String[] args) {

int a = 5, b = 5;

// Preincrement: ++a (increments first, then uses the value)

int pre = ++a;

System.out.println("Preincrement: a = " + a + ", pre = " + pre);

// Output: a = 6, pre = 6

// Postincrement: b++ (uses value first, then increments)

int post = b++;

System.out.println("Postincrement: b = " + b + ", post = " + post);

// Output: b = 6, post = 5

}

}

### ****Key Differences****

| **Type** | **When is it incremented?** | **Value used in expression** |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ++variable (preincrement) | Before evaluation | **New (incremented) value** |  |  |  |
| variable++ (postincrement) | After evaluation | **Old (original) value** |  |  |  |

**Exercise 3.15:**

### ****(a) Incorrect**** if-else ****Statement****

#### **Given Code (Incorrect)**

java

CopyEdit

if (age >= 65);

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

else

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

#### **Errors:**

1. **Semicolon (;) after if statement**
   * This **ends the if statement prematurely**, causing else to be misplaced.
2. **Mismatched quotation marks ("Age is less than 65)")**
   * The closing quotation mark is missing before the last parenthesis.

#### **Corrected Code**

java

CopyEdit

if (age >= 65) {

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

} else {

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

}

### ****(b) Incorrect**** while ****Loop****

#### **Given Code (Incorrect)**

java

CopyEdit

int x = 1, total;

while (x <= 10) {

total += x;

++x;

}

#### **Errors:**

1. **Uninitialized total**
   * The variable total is declared but **not initialized**, which will cause a compilation error.

#### **Corrected Code**

java

CopyEdit

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

while (x <= 10) {

total += x;

++x;

}

### ****(c) Incorrect**** while ****Loop****

#### **Given Code (Incorrect)**

java

CopyEdit

while (x <= 100)

total += x;

++x;

#### **Errors:**

1. **Missing braces {}**
   * Since while only applies to the **first statement**, ++x; will execute **outside the loop**, causing an infinite loop if x is never incremented.

#### **Corrected Code**

java

CopyEdit

while (x <= 100) {

total += x;

++x;

}

### ****(d) Incorrect**** while ****Loop****

#### **Given Code (Incorrect)**

java

CopyEdit

while (y > 0) {

System.out.println(y);

++y;

#### **Errors:**

1. **Missing closing brace }**
   * The while loop block **is not properly closed**.
2. **Infinite loop**
   * y is **increasing** (++y;), so if y > 0, it will never exit.

#### **Corrected Code**

java

CopyEdit

while (y > 0) {

System.out.println(y);

--y; // Decrement to prevent an infinite loop

}