**Chapter 3**

**Exercise 1**

3.10

The key difference is that an 'if' statement executes a block of code once if the condition is true, whereas a 'while' statement continues to execute the code repeatedly until the condition becomes false. 'If' is a control statement, while 'while' is a looping statement.

3.11

When a Java program divides one integer by another, it performs **integer division**, meaning the result is also an integer. Any decimal or fractional part is **discarded**, not rounded. For example, dividing 7 by 2 results in 3 instead of 3.5 because Java only keeps the whole number and removes the decimal portion.

This happens because integers in Java cannot store decimal values. When both numbers in a division operation are integers, Java assumes the result should also be an integer, even if mathematically it should have a fractional part.

To avoid this truncation and retain the decimal portion, at least one of the numbers must be a floating-point type, such as double or float. Java will then perform **floating-point division**, ensuring the result includes the decimal part instead of discarding it.

3.12

Control statements in Java can be combined in two ways:

1. **Sequential Combination:**  
   Statements are placed one after another and run in order. Each statement executes after the previous one finishes.
2. **Hierarchical Combination (Nesting):**  
   A control statement is placed inside another. For example, an if statement inside a for loop. This helps handle more complex conditions and repetitions.

These combinations help create smooth and logical program flows

3.13

**For loop for the first 100 integers; while loop for arbitrary integers.**

**Explanation**

1. **Identify the type of repetition for calculating the sum of the first 100 positive integers.**  
   For calculating the sum of the first 100 positive integers, a **for loop** would be appropriate. This is because the number of iterations is known in advance (100 iterations). The loop can iterate from 1 to 100, adding each integer to a cumulative sum.
2. **Describe how to perform the task using a for loop.**  
   In a programming context, the for loop can be structured as follows:
   1. Initialize a variable to hold the sum (e.g., sum = 0)
   2. Use a for loop that iterates from 1 to 100.
   3. In each iteration, add the current integer to the sum variable.
   4. After the loop, the sum variable will contain the total sum of the first 100 positive integers.
3. **Identify the type of repetition for calculating the sum of an arbitrary number of positive integers.**  
   For calculating the sum of an arbitrary number of positive integers, a **while loop** would be more appropriate. This is because the number of integers to sum is not predetermined, and the loop can continue until a specific condition is met (e.g., the user decides to stop inputting numbers)
4. **Describe how to perform the task using a while loop.**  
   In a programming context, the while loop can be structured as follows:
   1. Initialize a variable to hold the sum (e.g., sum = 0)
   2. Use a while loop that continues as long as the user wants to input numbers (e.g., while user\_input\_is\_valid)
   3. In each iteration, prompt the user to enter a positive integer and add it to the sum variable.
   4. After the loop ends (when the user decides to stop), the sum variable will contain the total sum of the entered positive integers.

#### **3.13**

#### **Type of Repetition for Sum Calculations**

* **Fixed Iteration (for-loop):** When summing the first 100 numbers, use for because the count is known.
* **Indeterminate Iteration (while-loop):** When summing an arbitrary number of inputs, use while until a sentinel value is encountered.

#### **3.14 Preincrement vs. Postincrement**

* **Preincrement (++x)**: Increases the value before using it.
* **Postincrement (x++)**: Uses the value first, then increments.

#### **3.15 Identifying and Fixing Errors**

**(a) Error: Semicolon after if and incorrect else syntax.**

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) Error: total is not initialized.**

java

CopyEdit

int x = 1, total = 0;

while (x <= 10) {

total += x;

++x;

}

**(c) Error: while loop missing braces; total update should be inside loop.**

java

CopyEdit

while (x <= 100) {

total += x;

++x;

}

**(d) Error: Infinite loop due to incrementing y in the wrong direction.**

java

CopyEdit

while (y > 0) {

System.out.println(y);

--y;

}

### ****Exercise 2 (Coding Questions)****

#### **3.17 Gas Mileage Calculator**

This program calculates miles per gallon (MPG) for each trip and an overall average.

java

CopyEdit

import java.util.Scanner;

public class GasMileage {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int miles, gallons;

double totalMiles = 0, totalGallons = 0;

while (true) {

System.out.print("Enter miles driven (-1 to stop): ");

miles = input.nextInt();

if (miles == -1) break;

System.out.print("Enter gallons used: ");

gallons = input.nextInt();

double mpg = (double) miles / gallons;

System.out.printf("Miles per gallon: %.2f%n", mpg);

totalMiles += miles;

totalGallons += gallons;

System.out.printf("Total MPG so far: %.2f%n", totalMiles / totalGallons);

}

input.close();

}

}

#### **3.18 Credit Limit Calculator**

java

CopyEdit

import java.util.Scanner;

public class CreditLimit {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter account number: ");

int accountNumber = input.nextInt();

System.out.print("Enter beginning balance: ");

int balance = input.nextInt();

System.out.print("Enter total charges: ");

int charges = input.nextInt();

System.out.print("Enter total credits: ");

int credits = input.nextInt();

System.out.print("Enter credit limit: ");

int creditLimit = input.nextInt();

balance = balance + charges - credits;

System.out.println("New balance: " + balance);

if (balance > creditLimit) {

System.out.println("Credit limit exceeded.");

}

input.close();

}

}

#### **3.19 Sales Commission Calculator**

java

CopyEdit

import java.util.Scanner;

public class SalesCommission {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double totalSales = 0;

while (true) {

System.out.print("Enter item value (-1 to stop): ");

double itemValue = input.nextDouble();

if (itemValue == -1) break;

totalSales += itemValue;

}

double earnings = 200 + (0.09 \* totalSales);

System.out.printf("Total earnings: $%.2f%n", earnings);

input.close();

}

}

#### **3.20 Salary Calculator**

java

CopyEdit

import java.util.Scanner;

public class SalaryCalculator {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

for (int i = 1; i <= 3; i++) {

System.out.printf("Enter hours worked for Employee %d: ", i);

int hours = input.nextInt();

System.out.print("Enter hourly rate: ");

double rate = input.nextDouble();

double salary = (hours <= 40) ? (hours \* rate) :

(40 \* rate + (hours - 40) \* 1.5 \* rate);

System.out.printf("Gross pay: $%.2f%n", salary);

}

input.close();

}

}

#### **3.21 Find the Largest Number (Pseudocode & Java Code)**

**Pseudocode:**

1. Initialize largest = Integer.MIN\_VALUE
2. Loop 10 times, input a number
3. If the number is greater than largest, update largest
4. Print largest

**Java Code:**

java

CopyEdit

import java.util.Scanner;

public class LargestNumber {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int largest = Integer.MIN\_VALUE, number;

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

System.out.print("Enter number: ");

number = input.nextInt();

if (number > largest) largest = number;

}

System.out.println("The largest number is: " + largest);

input.close();

}

}

### ****3.38 Cryptography Encryption & Decryption****

java

CopyEdit

import java.util.Scanner;

public class Cryptography {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter a 4-digit number: ");

int num = input.nextInt();

int d1 = (num / 1000 + 7) % 10;

int d2 = (num / 100 % 10 + 7) % 10;

int d3 = (num / 10 % 10 + 7) % 10;

int d4 = (num % 10 + 7) % 10;

int encrypted = d3 \* 1000 + d4 \* 100 + d1 \* 10 + d2;

System.out.println("Encrypted number: " + encrypted);

input.close();

}

}