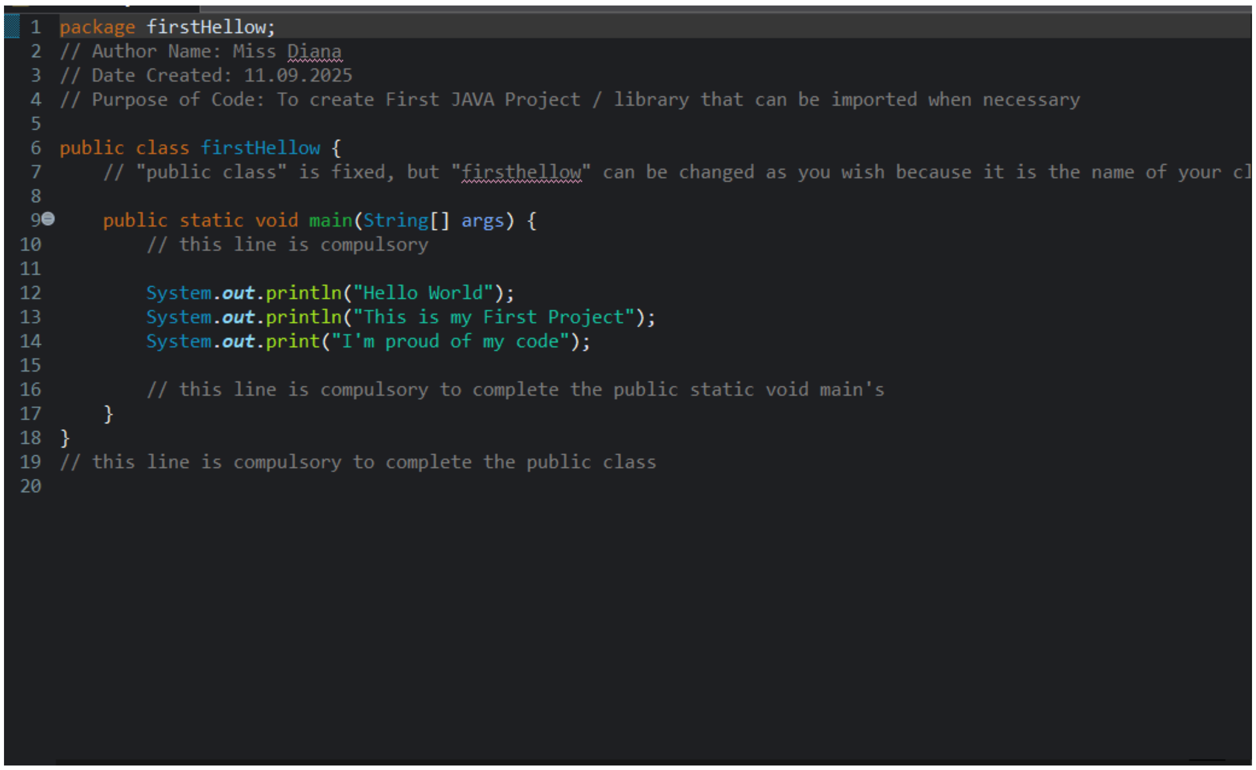
Lab Practical 3

PART A — Re-type Chapter 1 Code

Code 1: First Java Project:

Package: firsthellow

Class: firsthellow



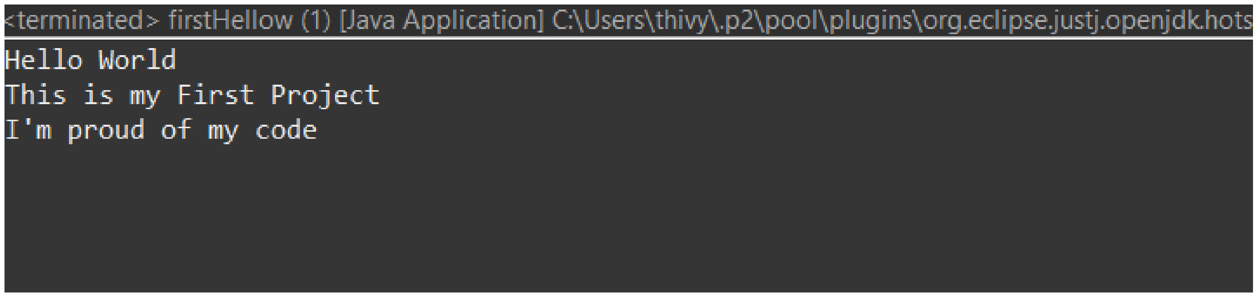
```java
package firstHellow;
// Author Name: Miss Diana
// Date Created: 11.09.2025
// Purpose of Code: To create First JAVA Project / library that can be imported when necessary

public class firstHellow {
    // "public class" is fixed, but "firsthellow" can be changed as you wish because it is the name of your cl

    public static void main(String[] args) {
        // this line is compulsory

        System.out.println("Hello World");
        System.out.println("This is my First Project");
        System.out.print("I'm proud of my code");

        // this line is compulsory to complete the public static void main's
    }
}
// this line is compulsory to complete the public class

```

Explanation (line by line):

Line	Explanation
package firsthellow;	Declares the package name for this Java file.
// Author Name: Miss Diana	Comment indicating the author of the program.

Line	Explanation
// Date Created: 11.09.2025	Specifies the date when this code was written.
// Purpose of Code: ...	Describes the purpose of the project.
public class firsthellow	Declares the main class for this program.
public static void main(String[] args)	The main method where execution begins.
System.out.println("Hello World");	Prints the text “Hello World” followed by a new line.
System.out.println("This is my First Project");	Prints another message on the next line.
System.out.print("I'm proud of my code");	Prints a message on the same line without adding a newline.
}	Closes the main method.
}	Closes the class definition.

Screenshot of the output:



```
<terminated> firstHellow (1) [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Hello World
This is my First Project
I'm proud of my code
```

Error Log Table:

File / Class	Date	Error Message	Cause	Fix
firsthellow.java	2025-10-08	None	N/A	Code executed successfully

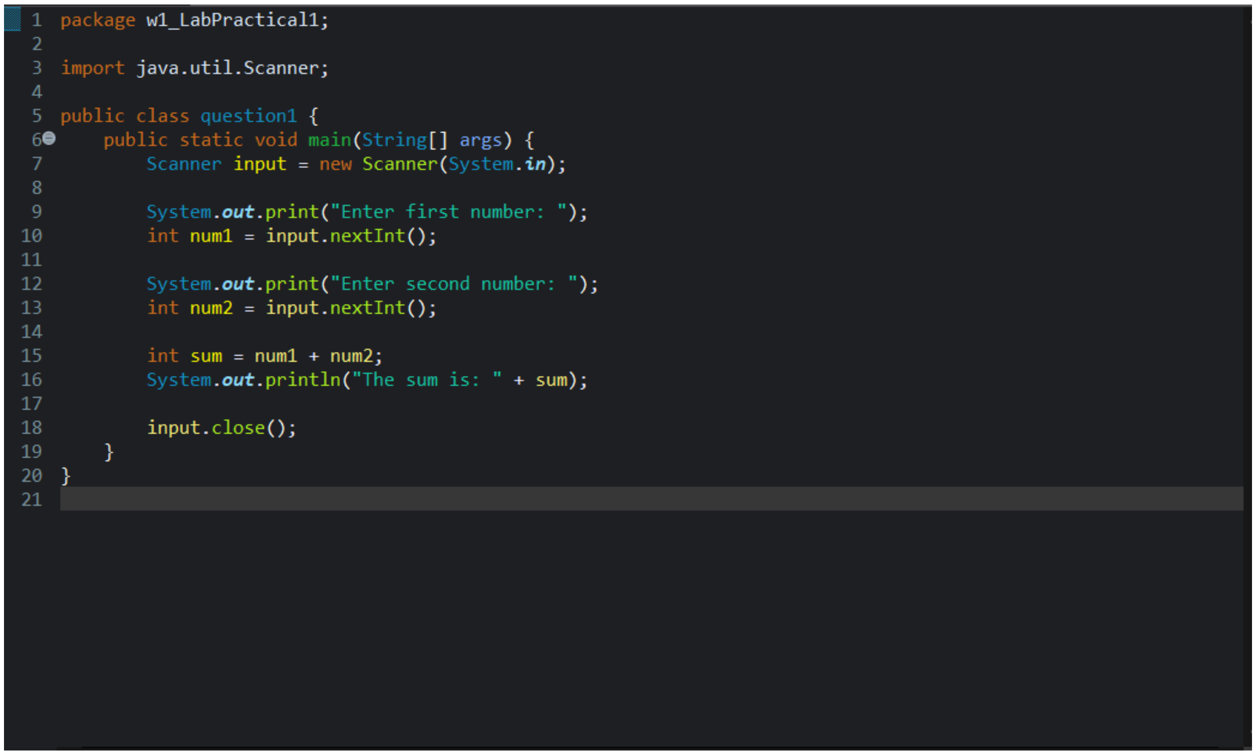
Self-Reflection:

I learned how to create my first Java program and understood the structure of a class and main method. I now know the difference between print and println. Before doing this exercise, my confidence was 4/10, and after completing it, it increased to 8/10.

Code 2: Basic Addition Program:

Package: w1_LabPractical1

Class: question1

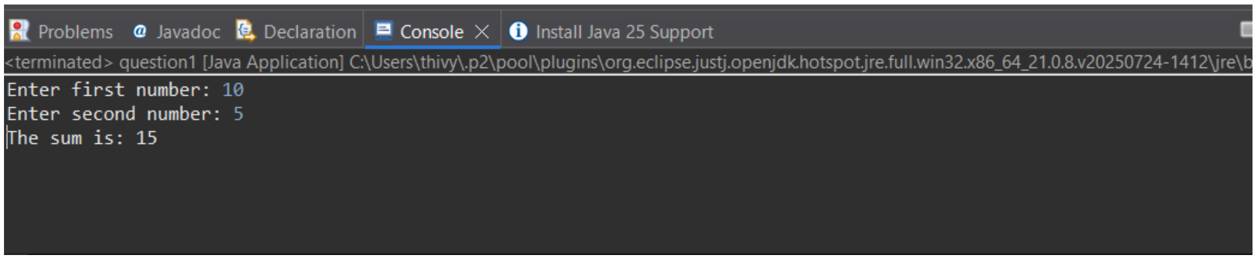


```java
package w1_LabPractical1;

import java.util.Scanner;

public class question1 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int num1 = input.nextInt();

        System.out.print("Enter second number: ");
        int num2 = input.nextInt();

        int sum = num1 + num2;
        System.out.println("The sum is: " + sum);

        input.close();
    }
}
```

Explanation (line by line):

Line	Explanation
package w1_LabPractical1;	Defines the package for this file.
import java.util.Scanner;	Imports the Scanner class for user input.
public class question1	Declares the class.
Scanner input = new Scanner(System.in);	Creates a Scanner object to take input.
System.out.print("Enter first number:");	Prompts user for the first number.
int num1 = input.nextInt();	Reads the first number.
System.out.print("Enter second number:");	Prompts for the second number.
int num2 = input.nextInt();	Reads the second number.
int sum = num1 + num2;	Adds both numbers.
System.out.println("The sum is: " + sum);	Displays the total.
input.close();	Closes the scanner.

Screenshot of the output:



```
<terminated> question1 [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\b
Enter first number: 10
Enter second number: 5
The sum is: 15
```

Error Log Table:

File / Class	Date	Error Message	Cause	Fix
question1.java	2025-10-08	None	N/A	Code executed successfully

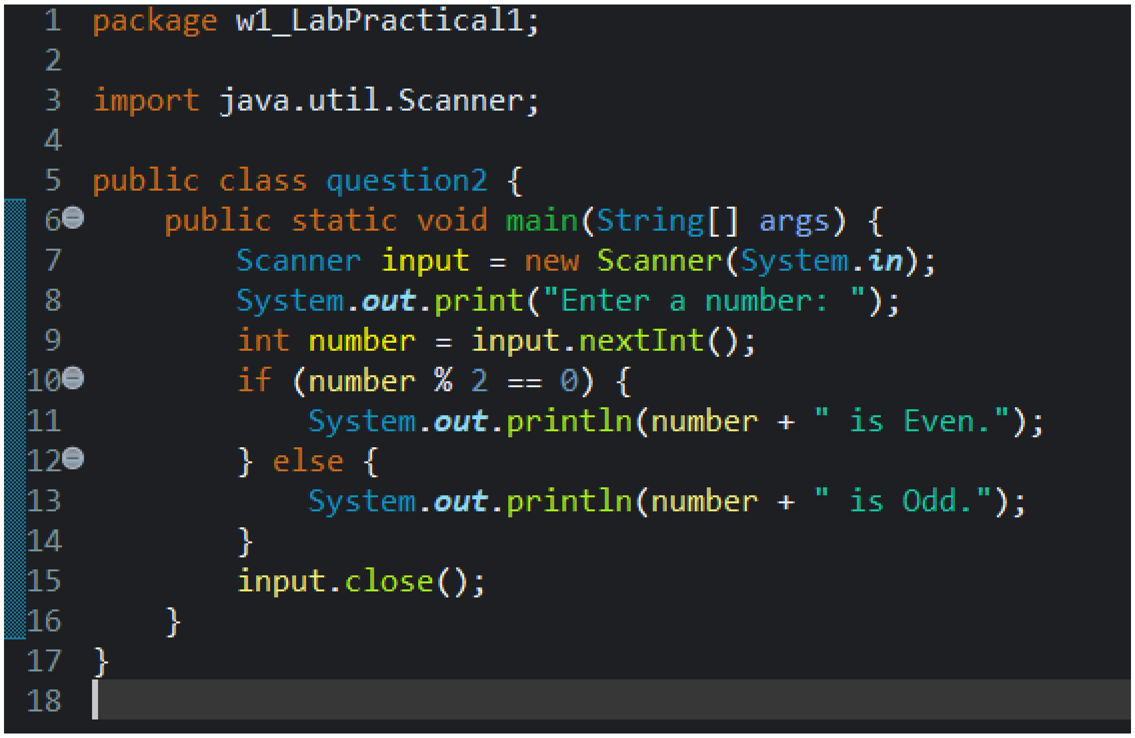
Self-Reflection:

This program helped me understand how to use the Scanner class to receive input from users and perform simple arithmetic operations. Before this exercise, my confidence level was 5/10; after completing it, it rose to 8/10.

Code 3: Even or Odd Number Checker:

Package: w1_LabPractical1

Class: question2

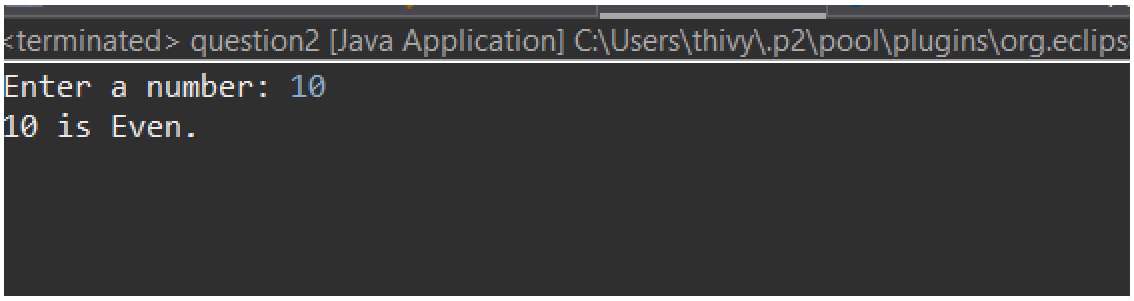


```java
package w1_LabPractical1;

import java.util.Scanner;

public class question2 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = input.nextInt();
        if (number % 2 == 0) {
            System.out.println(number + " is Even.");
        } else {
            System.out.println(number + " is Odd.");
        }
        input.close();
    }
}

```

Explanation (line by line):

Line	Explanation
Scanner input = new Scanner(System.in);	Creates a Scanner for user input.
System.out.print("Enter a number:");	Asks the user to input a number.
int number = input.nextInt();	Reads the input number.
if (number % 2 == 0)	Checks if the number is divisible by 2 (even).
System.out.println(number + " is Even.");	Displays message for even number.
else	Executes if the number is not even.
System.out.println(number + " is Odd.");	Displays message for odd number.
input.close();	Closes Scanner.

Screenshot of the output:



```
<terminated> question2 [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclips
Enter a number: 10
10 is Even.
```

Error Log Table:

File / Class	Date	Error Message	Cause	Fix
question2.java	2025-10-08	None	N/A	Code executed successfully

Self-Reflection:

I learned how to use conditional statements to check for even and odd numbers. I also practiced using the modulus operator (%). Before this exercise, my confidence was 6/10, and after finishing, it became 9/10.

PART B – Case Studies

Case Study 1: Smart Home Lighting System

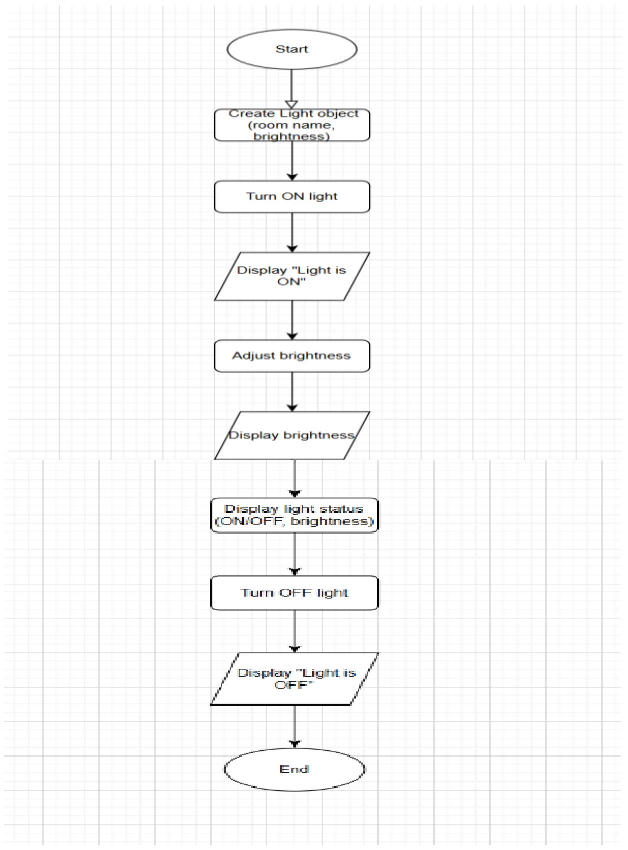
Problem Statement:

A family wants a Smart Home Lighting System. Each room has lights with different brightness.
The system should allow users to turn lights on/off, adjust brightness, and display the current status.
Different rooms may have different default brightness settings.

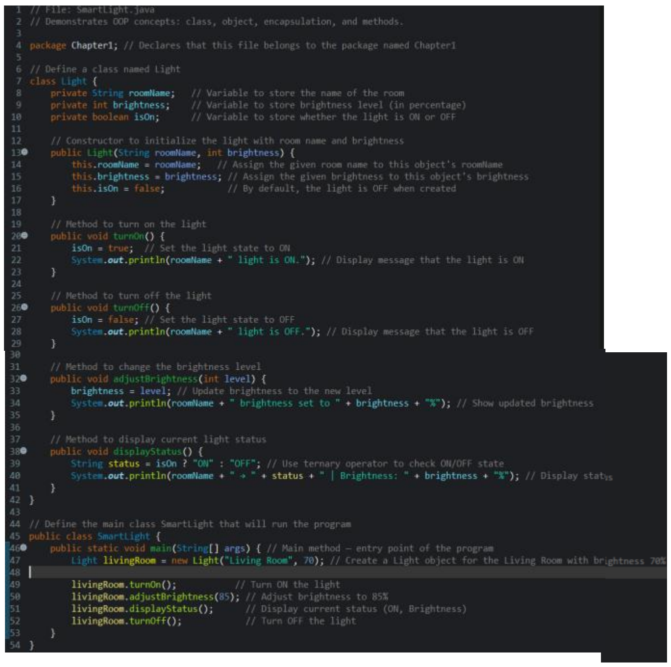
IPO Analysis:

Input	Process	Output
Room name, brightness value, on/off command	System updates brightness, toggles light status	Display of current light status and brightness

Flowchart:

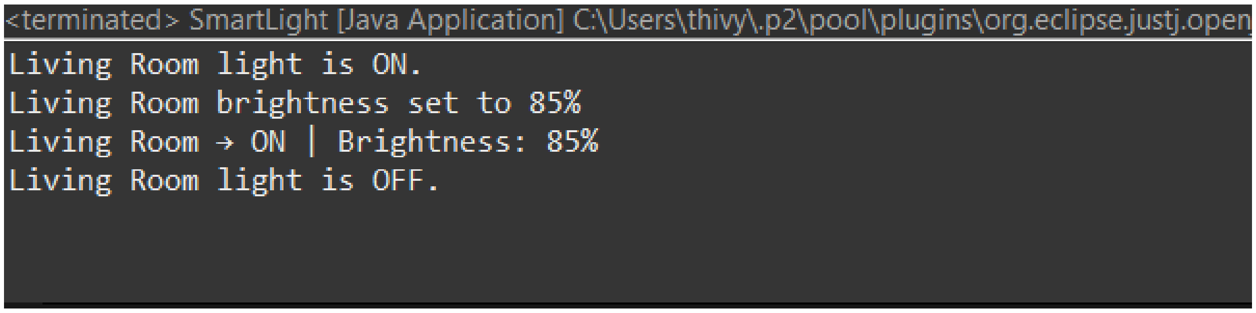


Source Code with Explanation:



```java
// File: SmartLight.java
// Demonstrates OOP concepts: class, object, encapsulation, and methods.

package Chapter1; // Declares that this file belongs to the package named Chapter1

// Define a class named Light
class Light {
    private String roomName;   // Variable to store the name of the room
    private int brightness;    // Variable to store brightness level (in percentage)
    private boolean isOn;      // Variable to store whether the light is ON or OFF

    // Constructor to initialize the light with room name and brightness
    public Light(String roomName, int brightness) {
        this.roomName = roomName;   // Assign the given room name to this object's roomName
        this.brightness = brightness; // Assign the given brightness to this object's brightness
        this.isOn = false;            // By default, the light is OFF when created
    }

    // Method to turn on the light
    public void turnOn() {
        isOn = true;  // Set the light state to ON
        System.out.println(roomName + " light is ON."); // Display message that the light is ON
    }

    // Method to turn off the light
    public void turnOff() {
        isOn = false; // Set the light state to OFF
        System.out.println(roomName + " light is OFF."); // Display message that the light is OFF
    }

    // Method to change the brightness level
    public void adjustBrightness(int level) {
        brightness = level; // Update brightness to the new level
        System.out.println(roomName + " brightness set to " + brightness + "%"); // Show updated brightness
    }

    // Method to display current light status
    public void displayStatus() {
        String status = isOn ? "ON" : "OFF"; // Use ternary operator to check ON/OFF state
        System.out.println(roomName + " → " + status + " | Brightness: " + brightness + "%"); // Display status
    }
}

// Define the main class SmartLight that will run the program
public class SmartLight {
    public static void main(String[] args) { // Main method – entry point of the program
        Light livingRoom = new Light("Living Room", 70); // Create a Light object for the Living Room with brightness 70%

        livingRoom.turnOn();           // Turn ON the light
        livingRoom.adjustBrightness(85); // Adjust brightness to 85%
        livingRoom.displayStatus();      // Display current status (ON, Brightness)
        livingRoom.turnOff();            // Turn OFF the light
    }
}
```

Output:


```
<terminated> SmartLight [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj.open
Living Room light is ON.
Living Room brightness set to 85%
Living Room → ON | Brightness: 85%
Living Room light is OFF.
```

Error Log Table:

Error	Description	Solution
Missing semicolon	Syntax error	Added ; at the end of the statement
Typo in variable name	Inconsistent variable usage	Corrected variable reference

References:

- W3Schools – Java Classes and Objects
- TutorialsPoint – Java Encapsulation

Self-Reflection:

This exercise helped me understand encapsulation and how to design a class with attributes and methods.
I learned to apply OOP to real-world systems like smart home automation.

Case Study 2: Online Food Ordering System

Problem Statement:

An online food delivery app allows users to order food items.

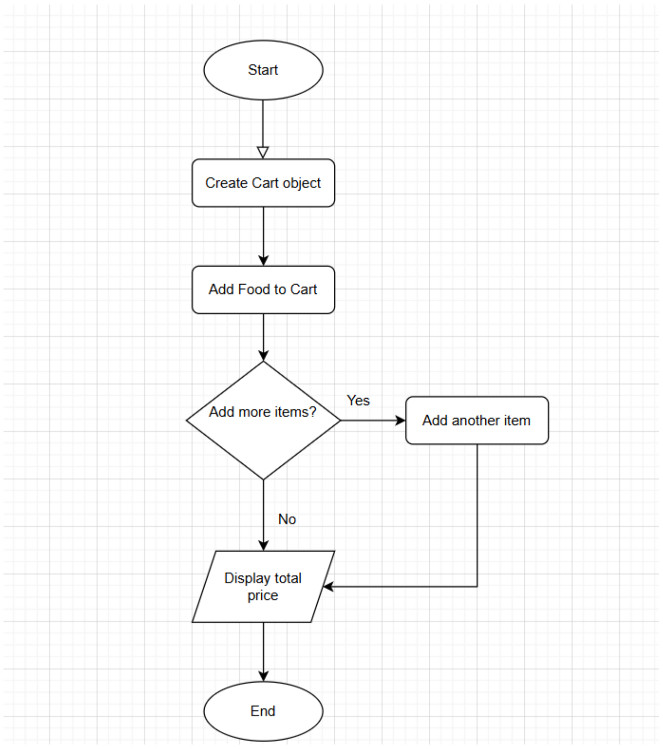
Each food has a name, price, and preparation time.

Customers can add food to a cart, view total price, and place an order.

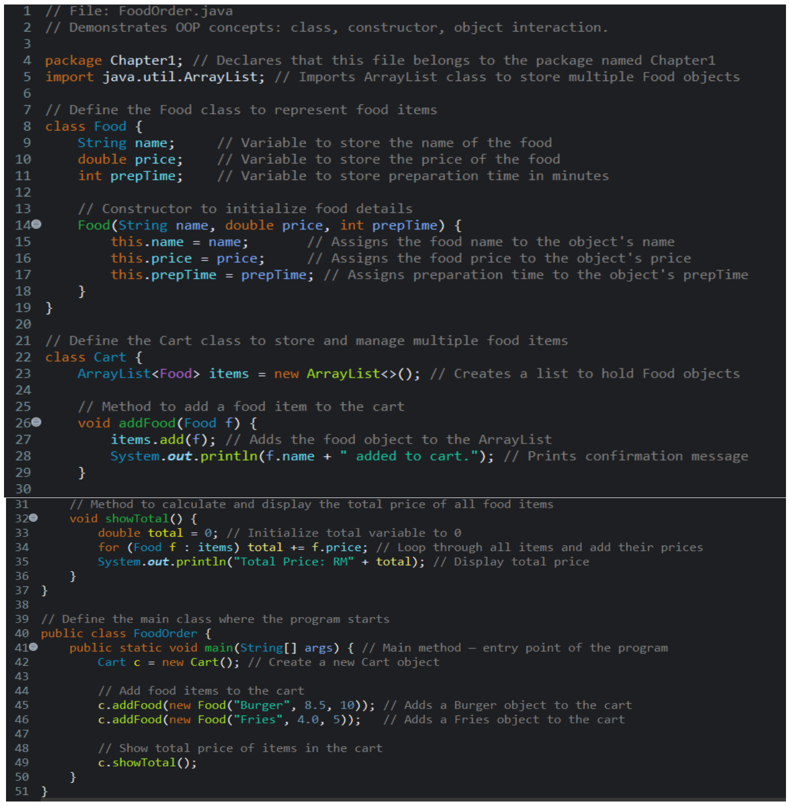
IPO Analysis:

Input	Process	Output
Food name, price, quantity	Add to cart, calculate total, display receipt	Total price and order confirmation

Flowchart:

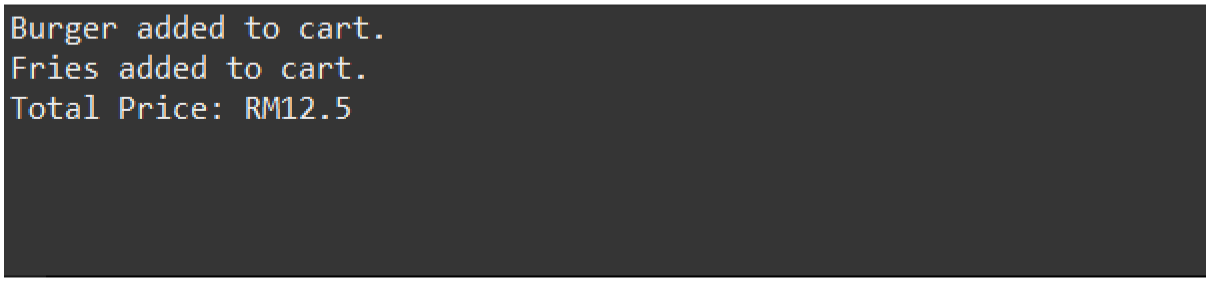


Source Code with Explanation:



```java
// File: FoodOrder.java
// Demonstrates OOP concepts: class, constructor, object interaction.

package Chapter1; // Declares that this file belongs to the package named Chapter1
import java.util.ArrayList; // Imports ArrayList class to store multiple Food objects

// Define the Food class to represent food items
class Food {
    String name;     // Variable to store the name of the food
    double price;    // Variable to store the price of the food
    int prepTime;    // Variable to store preparation time in minutes

    // Constructor to initialize food details
    Food(String name, double price, int prepTime) {
        this.name = name;        // Assigns the food name to the object's name
        this.price = price;      // Assigns the food price to the object's price
        this.prepTime = prepTime; // Assigns preparation time to the object's prepTime
    }
}

// Define the Cart class to store and manage multiple food items
class Cart {
    ArrayList<Food> items = new ArrayList<>(); // Creates a list to hold Food objects

    // Method to add a food item to the cart
    void addFood(Food f) {
        items.add(f); // Adds the food object to the ArrayList
        System.out.println(f.name + " added to cart."); // Prints confirmation message
    }

    // Method to calculate and display the total price of all food items
    void showTotal() {
        double total = 0; // Initialize total variable to 0
        for (Food f : items) total += f.price; // Loop through all items and add their prices
        System.out.println("Total Price: RM" + total); // Display total price
    }
}

// Define the main class where the program starts
public class FoodOrder {
    public static void main(String[] args) { // Main method – entry point of the program
        Cart c = new Cart(); // Create a new Cart object

        // Add food items to the cart
        c.addFood(new Food("Burger", 8.5, 10)); // Adds a Burger object to the cart
        c.addFood(new Food("Fries", 4.0, 5));   // Adds a Fries object to the cart

        // Show total price of items in the cart
        c.showTotal();
    }
}
```

Output:



```
Burger added to cart.
Fries added to cart.
Total Price: RM12.5
```

Error Log Table:

Error	Description	Solution
ArrayList not imported	Missing library	Added import java.util.ArrayList;

References:

- Java Docs – ArrayList Class
- TutorialsPoint – Object Interaction

Self-Reflection:

This task strengthened my understanding of class relationships and how lists manage multiple objects.

Case Study 3: Fitness Tracker Application

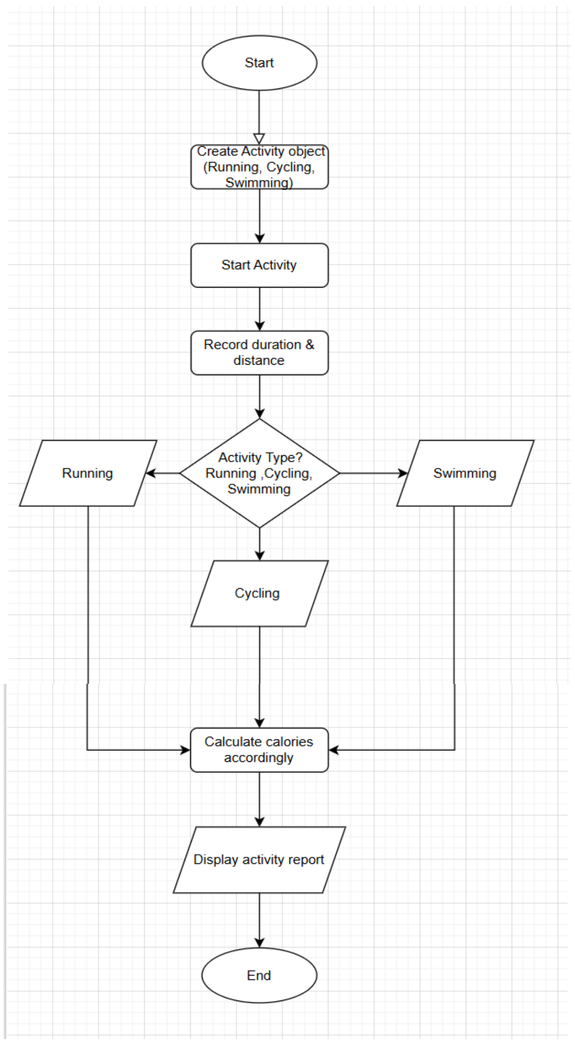
Problem Statement

A fitness tracker monitors activities such as running, cycling, and swimming.
Each activity records duration, distance, and calories burned.
Calories are calculated differently for each activity.

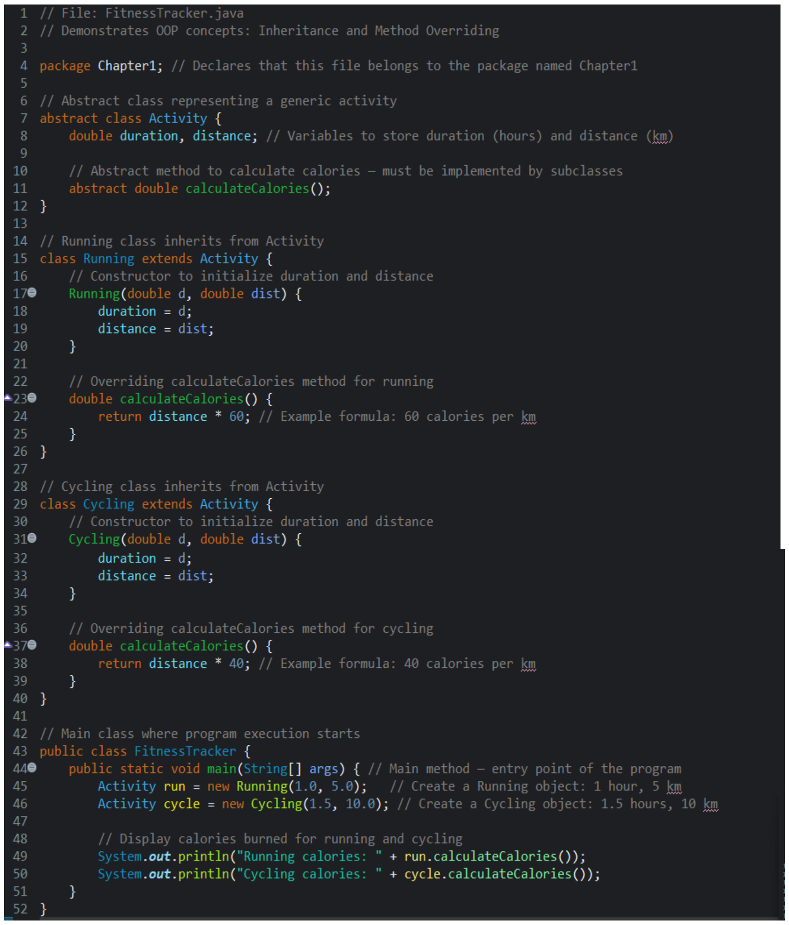
IPO Analysis:

Input	Process	Output
Activity type, duration, distance	Calculate calories based on activity type	Activity report

Flowchart:

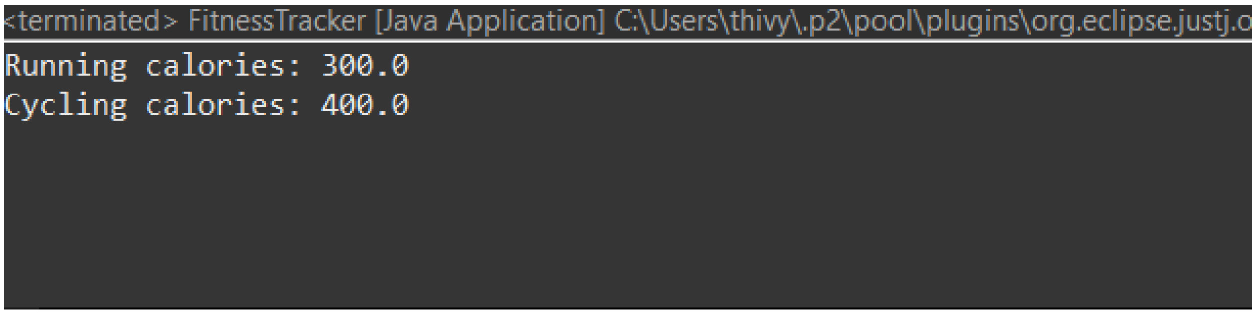

Start
Create Activity object (Running, Cycling, Swimming)
Start Activity
Record duration & distance
Activity Type? Running ,Cycling, Swimming
Running
Swimming
Cycling
Calculate calories accordingly
Display activity report
End

Source Code with Explanation:



```java
// File: FitnessTracker.java
// Demonstrates OOP concepts: Inheritance and Method Overriding

package Chapter1; // Declares that this file belongs to the package named Chapter1

// Abstract class representing a generic activity
abstract class Activity {
    double duration, distance; // Variables to store duration (hours) and distance (km)

    // Abstract method to calculate calories – must be implemented by subclasses
    abstract double calculateCalories();
}

// Running class inherits from Activity
class Running extends Activity {
    // Constructor to initialize duration and distance
    Running(double d, double dist) {
        duration = d;
        distance = dist;
    }

    // Overriding calculateCalories method for running
    double calculateCalories() {
        return distance * 60; // Example formula: 60 calories per km
    }
}

// Cycling class inherits from Activity
class Cycling extends Activity {
    // Constructor to initialize duration and distance
    Cycling(double d, double dist) {
        duration = d;
        distance = dist;
    }

    // Overriding calculateCalories method for cycling
    double calculateCalories() {
        return distance * 40; // Example formula: 40 calories per km
    }
}

// Main class where program execution starts
public class FitnessTracker {
    public static void main(String[] args) { // Main method – entry point of the program
        Activity run = new Running(1.0, 5.0);   // Create a Running object: 1 hour, 5 km
        Activity cycle = new Cycling(1.5, 10.0); // Create a Cycling object: 1.5 hours, 10 km

        // Display calories burned for running and cycling
        System.out.println("Running calories: " + run.calculateCalories());
        System.out.println("Cycling calories: " + cycle.calculateCalories());
    }
}
```

Output:



```
<terminated> FitnessTracker [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj.o
Running calories: 300.0
Cycling calories: 400.0
```

Error Log Table:

Error	Description	Solution
Abstract class instantiation	Illegal operation	Used subclass instead of abstract class

References:

- W3Schools – Java Inheritance
- Oracle Docs – Abstract Classes

Self-Reflection:

I learned to use inheritance and polymorphism for efficient code reuse and customization.

Case Study 4: E-Learning Quiz System

Problem Statement:

Each quiz has a title, number of questions, and difficulty level.

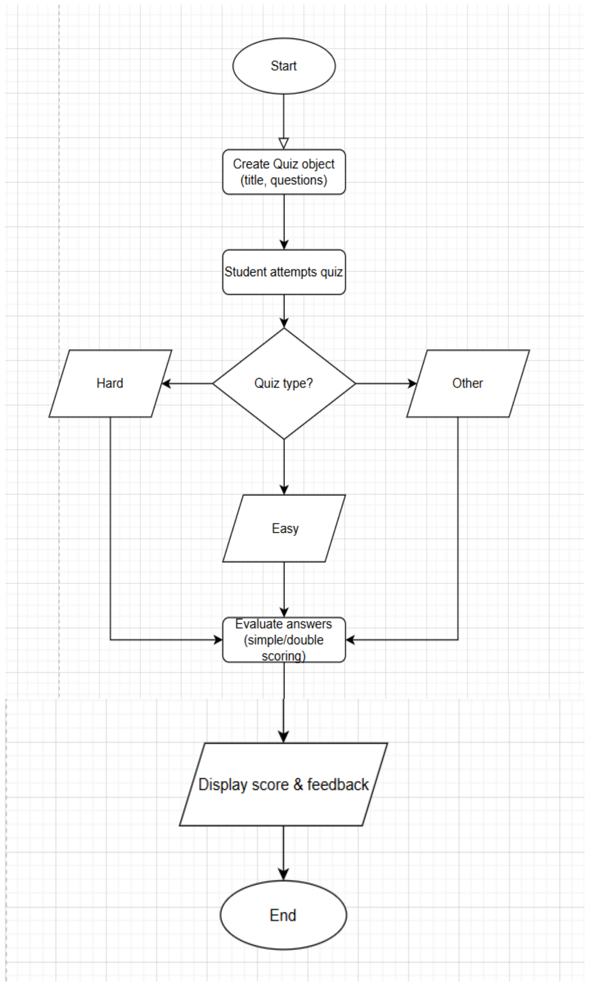
Students can attempt quizzes, submit answers, and receive scores.

Different quiz types evaluate answers differently.

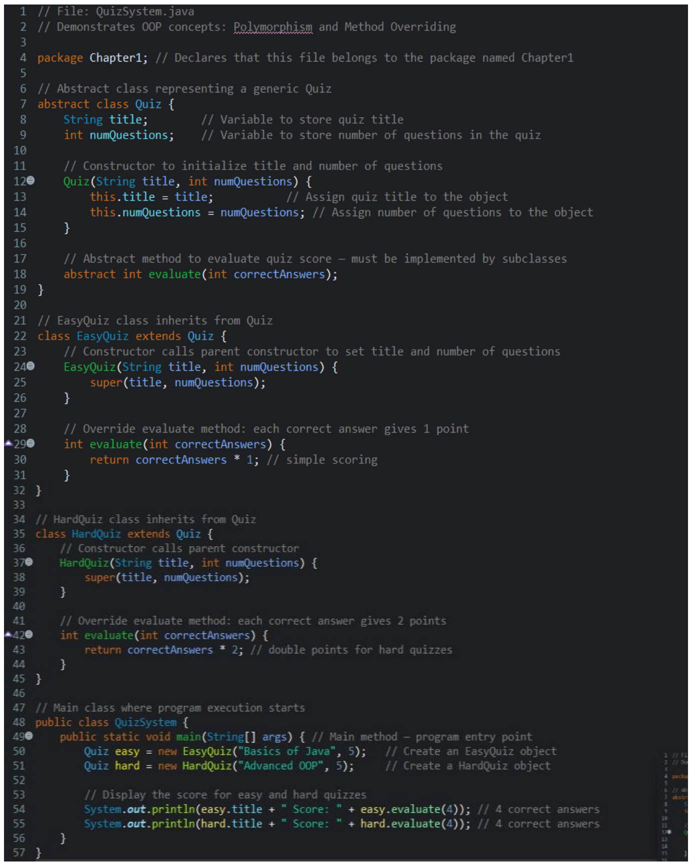
IPO Analysis:

Input	Process	Output
Quiz title, answers given	Evaluate answers, calculate score	Display score and feedback

Flowchart:

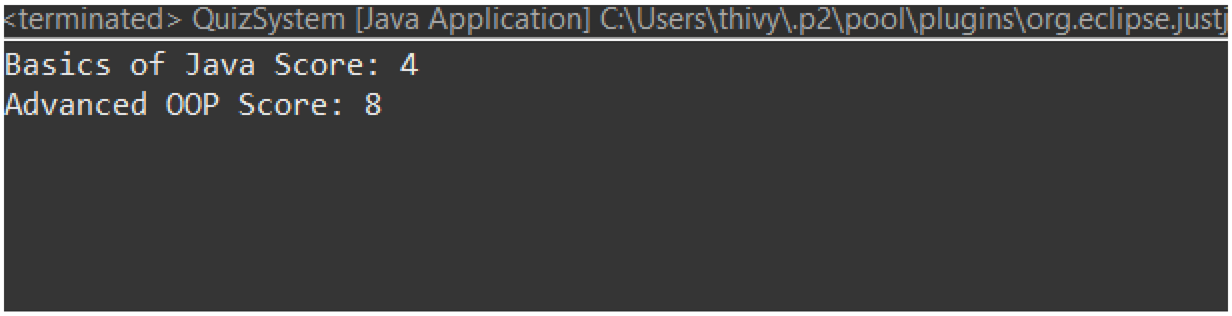

Start
Create Quiz object (title, questions)
Student attempts quiz
Quiz type?
Hard
Other
Easy
Evaluate answers (simple/double scoring)
Display score & feedback
End

Source Code with Explanation:



```java
// File: QuizSystem.java
// Demonstrates OOP concepts: Polymorphism and Method Overriding

package Chapter1; // Declares that this file belongs to the package named Chapter1

// Abstract class representing a generic Quiz
abstract class Quiz {
    String title;        // Variable to store quiz title
    int numQuestions;    // Variable to store number of questions in the quiz

    // Constructor to initialize title and number of questions
    Quiz(String title, int numQuestions) {
        this.title = title;             // Assign quiz title to the object
        this.numQuestions = numQuestions; // Assign number of questions to the object
    }

    // Abstract method to evaluate quiz score – must be implemented by subclasses
    abstract int evaluate(int correctAnswers);
}

// EasyQuiz class inherits from Quiz
class EasyQuiz extends Quiz {
    // Constructor calls parent constructor to set title and number of questions
    EasyQuiz(String title, int numQuestions) {
        super(title, numQuestions);
    }

    // Override evaluate method: each correct answer gives 1 point
    int evaluate(int correctAnswers) {
        return correctAnswers * 1; // simple scoring
    }
}

// HardQuiz class inherits from Quiz
class HardQuiz extends Quiz {
    // Constructor calls parent constructor
    HardQuiz(String title, int numQuestions) {
        super(title, numQuestions);
    }

    // Override evaluate method: each correct answer gives 2 points
    int evaluate(int correctAnswers) {
        return correctAnswers * 2; // double points for hard quizzes
    }
}

// Main class where program execution starts
public class QuizSystem {
    public static void main(String[] args) { // Main method – program entry point
        Quiz easy = new EasyQuiz("Basics of Java", 5);   // Create an EasyQuiz object
        Quiz hard = new HardQuiz("Advanced OOP", 5);     // Create a HardQuiz object

        // Display the score for easy and hard quizzes
        System.out.println(easy.title + " Score: " + easy.evaluate(4)); // 4 correct answers
        System.out.println(hard.title + " Score: " + hard.evaluate(4)); // 4 correct answers
    }
}
```

Output:



```
<terminated> QuizSystem [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj
Basics of Java Score: 4
Advanced OOP Score: 8
```

Error Log Table:

Error	Description	Solution
Constructor error	Missing super()	Added call to parent constructor

References:

- Java Docs – Polymorphism
- W3Schools – Inheritance and Overriding

Self-Reflection:

This helped me practice polymorphism and scoring logic differences in derived classes.

Case Study 5: Movie Ticket Booking System

Problem Statement:

A cinema allows users to book tickets for movies.

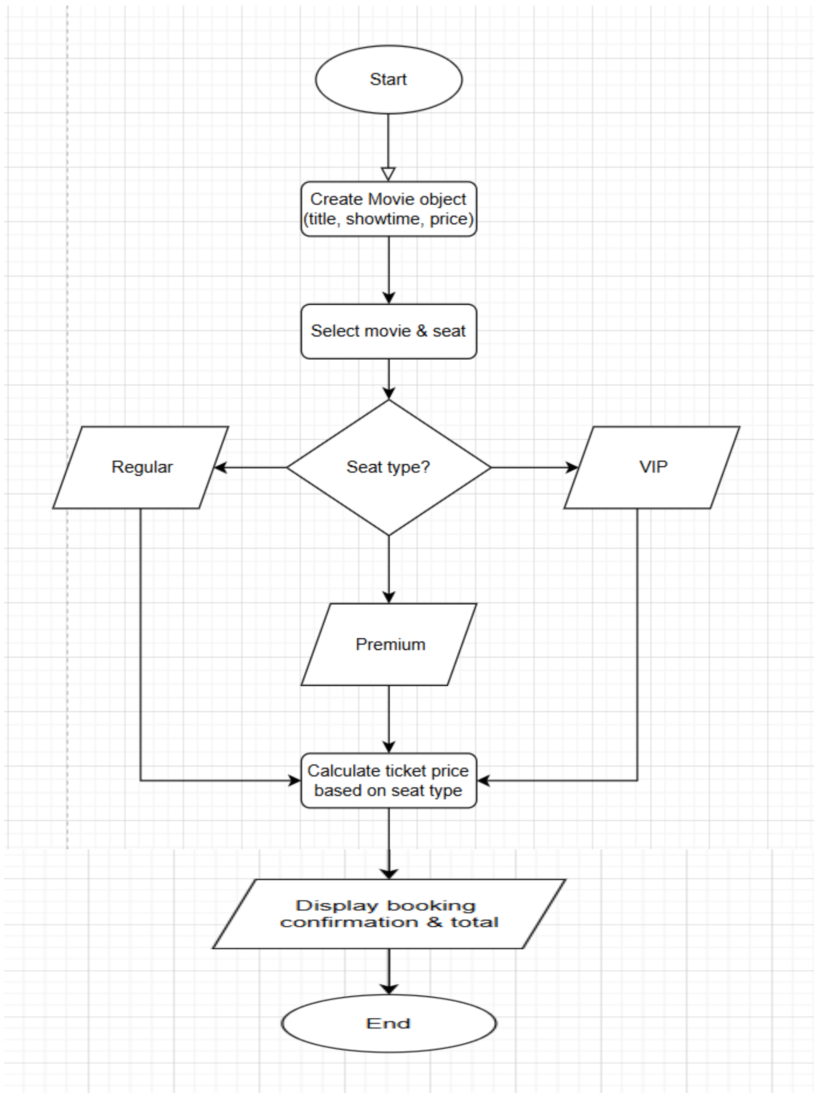
Each movie has a title, showtime, and ticket price.

Users can select a movie, choose seats, and confirm payment.

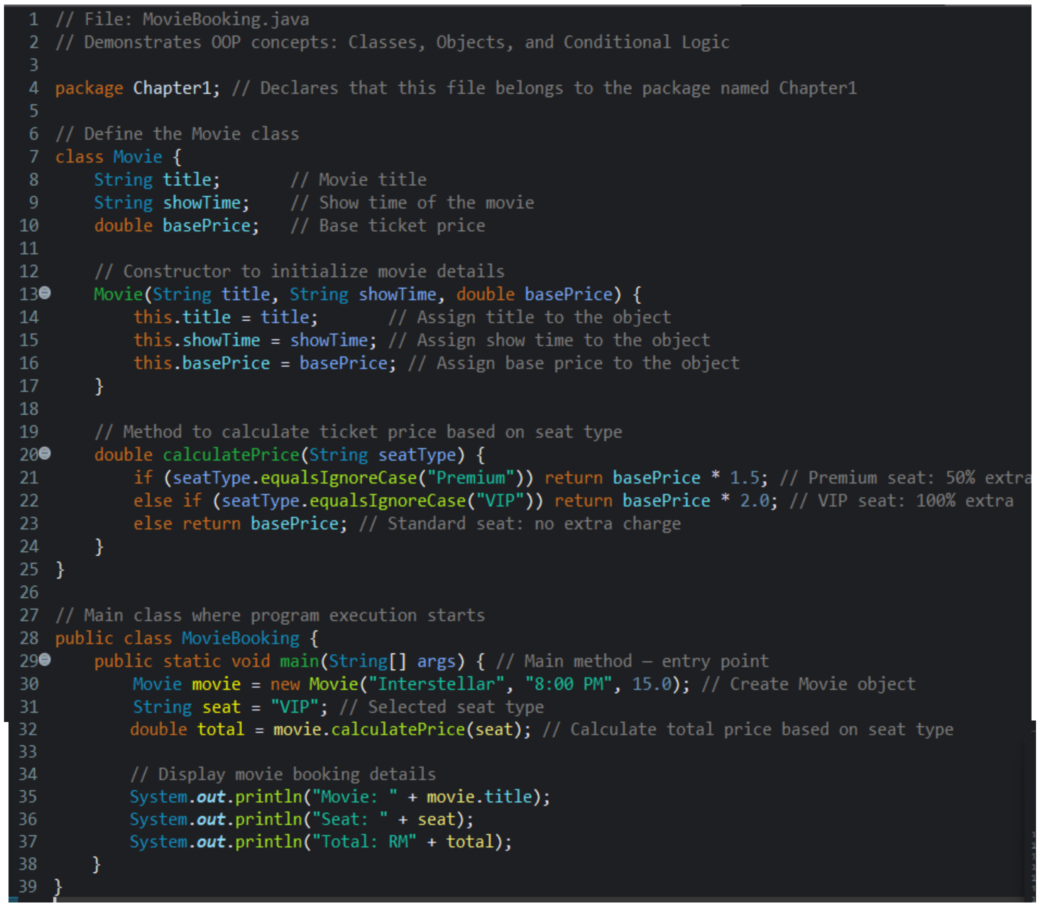
IPO Analysis:

Input	Process	Output
Movie title, seat type	Calculate price and confirm booking	Booking confirmation and total cost

Flowchart:

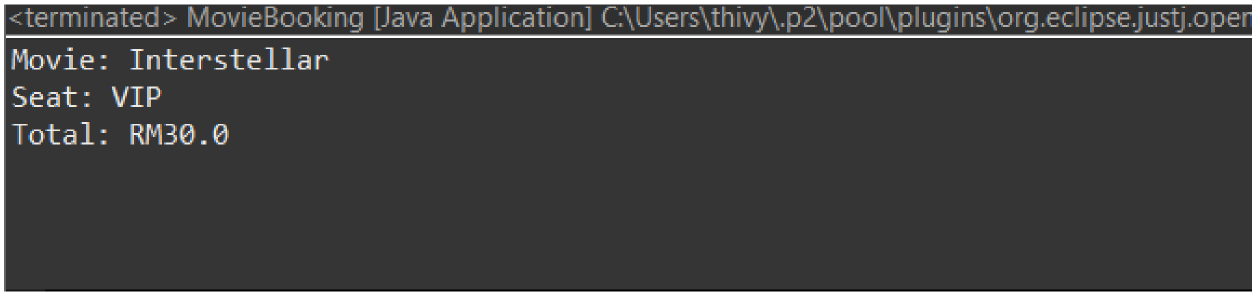

Start
Create Movie object (title, showtime, price)
Select movie & seat
Seat type?
Regular
VIP
Premium
Calculate ticket price based on seat type
Display booking confirmation & total
End

Source Code with Explanation:



```java
// File: MovieBooking.java
// Demonstrates OOP concepts: Classes, Objects, and Conditional Logic

package Chapter1; // Declares that this file belongs to the package named Chapter1

// Define the Movie class
class Movie {
    String title;       // Movie title
    String showTime;    // Show time of the movie
    double basePrice;   // Base ticket price

    // Constructor to initialize movie details
    Movie(String title, String showTime, double basePrice) {
        this.title = title;         // Assign title to the object
        this.showTime = showTime; // Assign show time to the object
        this.basePrice = basePrice; // Assign base price to the object
    }

    // Method to calculate ticket price based on seat type
    double calculatePrice(String seatType) {
        if (seatType.equalsIgnoreCase("Premium")) return basePrice * 1.5; // Premium seat: 50% extra
        else if (seatType.equalsIgnoreCase("VIP")) return basePrice * 2.0; // VIP seat: 100% extra
        else return basePrice; // Standard seat: no extra charge
    }
}

// Main class where program execution starts
public class MovieBooking {
    public static void main(String[] args) { // Main method – entry point
        Movie movie = new Movie("Interstellar", "8:00 PM", 15.0); // Create Movie object
        String seat = "VIP"; // Selected seat type
        double total = movie.calculatePrice(seat); // Calculate total price based on seat type

        // Display movie booking details
        System.out.println("Movie: " + movie.title);
        System.out.println("Seat: " + seat);
        System.out.println("Total: RM" + total);
    }
}
```

Output:



```
<terminated> MovieBooking [Java Application] C:\Users\thivy\.p2\pool\plugins\org.eclipse.justj.open
Movie: Interstellar
Seat: VIP
Total: RM30.0
```

Error Log Table:

Error	Description	Solution
Missing return value	Method missing return	Added return statement

References:

- Java Conditional Statements – W3Schools
- TutorialsPoint – Java Classes and Methods

Self-Reflection:

Through this, I learned to implement real-world logic using conditionals and OOP class structures.

Overall Reflection for Part B:

Completing all five case studies deepened my understanding of object-oriented programming. Before doing this practical, my OOP knowledge rating was 6 / 10; after completing all tasks, I rate myself 9 / 10 in confidence with Java OOP.

Link for JAVA codes in Github:

raam34567/DIT1334_Coursework