EXPERIMENT – 1

AIM:

Design a **console-based bus seat reservation system** for 10 seats. When a seat is already reserved, display an appropriate message.

PSEUDOCODE:

```
BEGIN
  CREATE an array called seats[10] of boolean values, initialized to false (meaning all seats are free)

  LOOP infinitely
    DISPLAY "--- Bus Reservation Menu ---"
    DISPLAY "1. Reserve Seat"
    DISPLAY "2. Display Seat Availability"
    DISPLAY "3. Exit"
    PROMPT user to enter an option
    READ option

    SWITCH (option)
      CASE 1:
        PROMPT "Enter seat number (1–10): "
        READ seatNum
        IF seatNum < 1 OR seatNum > 10 THEN
          DISPLAY "Invalid seat number!"
        ELSE IF seats[seatNum - 1] == true THEN
          DISPLAY "Seat already reserved!"
```

```
            ELSE
                SET seats[seatNum - 1] = true
                DISPLAY "Seat " + seatNum + " reserved successfully!"
            ENDIF
            BREAK

        CASE 2:
            DISPLAY "Seat Availability:"
            FOR i FROM 0 TO 9 DO
                IF seats[i] == true THEN
                    DISPLAY "Seat " + (i + 1) + ": Reserved"
                ELSE
                    DISPLAY "Seat " + (i + 1) + ": Available"
                ENDIF
            ENDFOR
            BREAK

        CASE 3:
            DISPLAY "Thank you for using the Bus Reservation System!"
            TERMINATE program

        DEFAULT:
            DISPLAY "Invalid option! Please try again."
    ENDSWITCH
  ENDLOOP
END
```

PROGRAM:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        boolean[] seats = new boolean[10];

        while (true) {
            System.out.println("\n--- Bus Reservation Menu ---");
            System.out.println("1. Reserve Seat");
            System.out.println("2. Display Seat Availability");
            System.out.println("3. Exit");
            System.out.print("Enter your option: ");

            int choice = input.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter seat number (1–10): ");
                    int seatNum = input.nextInt();
                    if (seatNum < 1 || seatNum > 10)
                        System.out.println("Invalid seat number!");
                    else if (seats[seatNum - 1])
                        System.out.println("Seat already reserved!");
```

```
                else {
                    seats[seatNum - 1] = true;
                    System.out.println("Seat " + seatNum + " reserved successfully!");
                }
                break;

            case 2:
                System.out.println("\nSeat Availability:");
                for (int i = 0; i < 10; i++) {
                    System.out.println("Seat " + (i + 1) + ": " + (seats[i] ? "Reserved" :
"Available"));
                }
                break;

            case 3:
                System.out.println("Thank you for using the Bus Reservation System!");
                return;

            default:
                System.out.println("Invalid option! Please try again.");
        }
    }
  }
}
```

PROGRAM LOGIC:

1) The program starts with the main method.
2) A Scanner object is created to take input from the user.
3) A boolean array seats[10] is initialized to represent the reservation status of each seat (false = available, true = reserved).
4) A continuous loop displays the reservation menu.
5) The user selects one of three options:
    - **Option 1:** Reserve a seat.
        - The seat number is entered and validated (1–10).
        - If the seat is already reserved, a message is shown.
        - Otherwise, it's marked as reserved.
    - **Option 2:** Display seat status.
        - The system lists all seats with either "Reserved" or "Available" beside them.
    - **Option 3:** Exit the application.
6) Invalid menu choices are handled gracefully.
7) The program repeats until the user exits.

OUTPUT:

```
--- Bus Reservation Menu ---
1. Reserve Seat
2. Display Seat Availability
3. Exit
Enter your option: 1
Enter seat number (1?10): 3
Seat 3 reserved successfully!

--- Bus Reservation Menu ---
1. Reserve Seat
2. Display Seat Availability
3. Exit
Enter your option: 1
Enter seat number (1?10): 3
Seat already reserved!

--- Bus Reservation Menu ---
1. Reserve Seat
2. Display Seat Availability
3. Exit
Enter your option: 2
```

```
Seat Availability:
Seat 1: Available
Seat 2: Available
Seat 3: Reserved
Seat 4: Available
Seat 5: Available
Seat 6: Available
Seat 7: Available
Seat 8: Available
Seat 9: Available
Seat 10: Available

--- Bus Reservation Menu ---
1. Reserve Seat
2. Display Seat Availability
3. Exit
Enter your option: 3
Thank you for using the Bus Reservation System!

=== Code Execution Successful ===
```

CONCLUSION:

This experiment demonstrates how to manage seat reservations in a console-based environment using arrays and conditional checks. It enhances understanding of loops, decision-making, and array manipulation, laying the groundwork for larger real-world systems like bus or train booking applications. Which has been done and executed successfully.

EXPERIMENT – 2

AIM:

Develop a console-based banking application that allows a user to check their balance, deposit money, or withdraw funds.

PSEUDOCODE:

```
BEGIN
  SET balance = 1500.0
  CREATE Scanner object for user input

  LOOP infinitely
    DISPLAY "--- Banking Operations Menu ---"
    DISPLAY "1. View Balance"
    DISPLAY "2. Deposit Amount"
    DISPLAY "3. Withdraw Amount"
    DISPLAY "4. Exit"
    PROMPT user to enter choice
    READ choice

    SWITCH(choice)
      CASE 1:
        DISPLAY "Current Balance: ₹" + balance
        BREAK

      CASE 2:
        PROMPT "Enter deposit amount: "
        READ deposit
```

```
            ADD deposit to balance
            DISPLAY "Deposit successful!"
            BREAK

        CASE 3:
            PROMPT "Enter withdrawal amount: "
            READ withdrawal
            IF withdrawal > balance THEN
                DISPLAY "Insufficient balance!"
            ELSE
                SUBTRACT withdrawal from balance
                DISPLAY "Withdrawal successful!"
            ENDIF
            BREAK

        CASE 4:
            DISPLAY "Thank you for banking with us!"
            EXIT program

        DEFAULT:
            DISPLAY "Invalid option! Please select again."
    ENDSWITCH
  ENDLOOP
END
```

PROGRAM:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double balance = 1500.0;

        while (true) {
            System.out.println("\n--- Banking Operations Menu ---");
            System.out.println("1. View Balance");
            System.out.println("2. Deposit Amount");
            System.out.println("3. Withdraw Amount");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Current Balance: ₹" + balance);
                    break;
                case 2:
                    System.out.print("Enter deposit amount: ");
                    double deposit = sc.nextDouble();
                    balance += deposit;
                    System.out.println("Deposit successful!");
```

```
                break;
            case 3:
                System.out.print("Enter withdrawal amount: ");
                double withdraw = sc.nextDouble();
                if (withdraw > balance)
                    System.out.println("Insufficient balance!");
                else {
                    balance -= withdraw;
                    System.out.println("Withdrawal successful!");
                }
                break;
            case 4:
                System.out.println("Thank you for banking with us!");
                return;
            default:
                System.out.println("Invalid option! Please select again.");
        }
    }
}
```

PROGRAM LOGIC:

1) Start the main method.
2) Initialize balance to 1500.0.
3) Continuously show a menu with four options.
4) Based on the user's input:
   - 1: Display current balance.
   - 2: Ask and add deposit amount.
   - 3: Check and process withdrawal if sufficient funds exist.
   - 4: Exit the loop and end the program.
5) Repeat the loop until user exits.

OUTPUT:

```
--- Banking Operations Menu ---
1. View Balance
2. Deposit Amount
3. Withdraw Amount
4. Exit
Enter your choice: 1
Current Balance: ?1500.0

--- Banking Operations Menu ---
1. View Balance
2. Deposit Amount
3. Withdraw Amount
4. Exit
Enter your choice: 2
Enter deposit amount: 2500
Deposit successful!

--- Banking Operations Menu ---
1. View Balance
2. Deposit Amount
3. Withdraw Amount
4. Exit
```

```
Enter your choice: 1
Current Balance: ?4000.0

--- Banking Operations Menu ---
1. View Balance
2. Deposit Amount
3. Withdraw Amount
4. Exit
Enter your choice: 3
Enter withdrawal amount: 1000
Withdrawal successful!

--- Banking Operations Menu ---
1. View Balance
2. Deposit Amount
3. Withdraw Amount
4. Exit
Enter your choice: 4
Thank you for banking with us!

=== Code Execution Successful ===
```

CONCLUSION:

This program simulates basic financial transactions, teaching logical sequencing, arithmetic operations, and condition-based validation in console applications. Which has been done and executed successfully.

EXPERIMENT – 3

AIM:

Create a **vehicle hire system** for 3 bikes where each can be hired or returned by the user.

PSEUDOCODE:

```
BEGIN
  CREATE boolean array bikes[3] initialized to false (available)
  CREATE Scanner for user input

  LOOP forever
    DISPLAY "--- Vehicle Hire Menu ---"
    DISPLAY "1. Hire Bike"
    DISPLAY "2. Return Bike"
    DISPLAY "3. View Availability"
    DISPLAY "4. Exit"
    PROMPT for user choice
    READ choice

    IF choice == 1 THEN
      PROMPT "Enter bike number (1–3): "
      READ num
      IF num < 1 OR num > 3 THEN
        DISPLAY "Invalid bike number!"
      ELSE IF bikes[num-1] == true THEN
        DISPLAY "Bike already hired!"
      ELSE
```

```
            SET bikes[num-1] = true
            DISPLAY "Bike " + num + " hired successfully!"
        ENDIF

    ELSE IF choice == 2 THEN
        PROMPT "Enter bike number to return: "
        READ num
        IF num < 1 OR num > 3 THEN
            DISPLAY "Invalid bike number!"
        ELSE IF bikes[num-1] == false THEN
            DISPLAY "This bike was not hired!"
        ELSE
            SET bikes[num-1] = false
            DISPLAY "Bike returned successfully!"
        ENDIF

    ELSE IF choice == 3 THEN
        FOR i FROM 0 TO 2 DO
            DISPLAY "Bike " + (i+1) + ": " + (bikes[i] ? "Hired" : "Available")
        ENDFOR

    ELSE IF choice == 4 THEN
        DISPLAY "Thank you for using Vehicle Hire System!"
        EXIT loop
    ENDIF
  ENDLOOP
END
```

PROGRAM:

```
import java.util.*;

public class Main{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      boolean[] bikes = new boolean[3];

      while (true) {
         System.out.println("\n--- Vehicle Hire Menu ---");
         System.out.println("1. Hire Bike");
         System.out.println("2. Return Bike");
         System.out.println("3. View Availability");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");

         int ch = sc.nextInt();

         if (ch == 1) {
            System.out.print("Enter bike number (1–3): ");
            int num = sc.nextInt();
            if (num < 1 || num > 3)
               System.out.println("Invalid bike number!");
            else if (bikes[num - 1])
               System.out.println("Bike already hired!");
            else {
               bikes[num - 1] = true;
               System.out.println("Bike " + num + " hired successfully!");
```

```
            }
        } else if (ch == 2) {
            System.out.print("Enter bike number to return: ");
            int num = sc.nextInt();
            if (num < 1 || num > 3)
                System.out.println("Invalid bike number!");
            else if (!bikes[num - 1])
                System.out.println("This bike was not hired!");
            else {
                bikes[num - 1] = false;
                System.out.println("Bike returned successfully!");
            }
        } else if (ch == 3) {
            for (int i = 0; i < 3; i++) {
                System.out.println("Bike " + (i + 1) + ": " + (bikes[i] ? "Hired" :
"Available"));
            }
        } else if (ch == 4) {
            System.out.println("Thank you for using Vehicle Hire System!");
            break;
        } else {
            System.out.println("Invalid option!");
        }
    }
}
```

PROGRAM LOGIC:

1)Declare a boolean array bikes[3] representing bike status.

2)Show a repeating menu with four actions.

3)Validate user input for bike number (1–3).

4)Modify array values based on hire/return operations.

5)Continue until user selects exit.

OUTPUT:

```
--- Vehicle Hire Menu ---
1. Hire Bike
2. Return Bike
3. View Availability
4. Exit
Enter your choice: 1
Enter bike number (1?3): 3
Bike 3 hired successfully!

--- Vehicle Hire Menu ---
1. Hire Bike
2. Return Bike
3. View Availability
4. Exit
Enter your choice: 1
Enter bike number (1?3): 2
Bike 2 hired successfully!
```

```
--- Vehicle Hire Menu ---
1. Hire Bike
2. Return Bike
3. View Availability
4. Exit
Enter your choice: 2
Enter bike number to return: 3
Bike returned successfully!

--- Vehicle Hire Menu ---
1. Hire Bike
2. Return Bike
3. View Availability
4. Exit
Enter your choice: 3
Bike 1: Available
Bike 2: Hired
Bike 3: Available
```

```
--- Vehicle Hire Menu ---
1. Hire Bike
2. Return Bike
3. View Availability
4. Exit
Enter your choice: 4
Thank you for using Vehicle Hire System!

=== Code Execution Successful ===
```

CONCLUSION:

This system simulates state-based resource allocation and management. It strengthens logical reasoning in handling conditions, loops, and data persistence across iterations. Which has been done and executed successfully.

EXPERIMENT – 4

AIM:

Implement a **Student Record Management System** using a class and ArrayList to store and display student details.

PSEUDOCODE:

```
BEGIN
  DEFINE class Student with fields name, id
  CREATE ArrayList<Student> list
  CREATE Scanner for user input

  LOOP forever
    DISPLAY "1. Add Student"
    DISPLAY "2. View Students"
    DISPLAY "3. Exit"
    PROMPT for user choice
    READ choice

    IF choice == 1 THEN
      PROMPT "Enter student name: "
      READ name
      PROMPT "Enter student ID: "
      READ id
      CREATE new Student(name, id)
      ADD Student to list
```

```
        DISPLAY "Student added successfully!"
    ELSE IF choice == 2 THEN
        DISPLAY "---- Student List ----"
        FOR each student in list
            DISPLAY name and id
        ENDFOR

    ELSE IF choice == 3 THEN
        DISPLAY "Thank you!"
        EXIT loop
    ENDIF
  ENDLOOP
END
```

PROGRAM:

```
import java.util.*;

class Student {
    String name;
    String id;

    Student(String name, String id) {
        this.name = name;
        this.id = id;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<Student> records = new ArrayList<>();

        while (true) {
            System.out.println("\n1. Add Student");
            System.out.println("2. View Students");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");

            int choice = sc.nextInt();
            sc.nextLine();
```

```
            if (choice == 1) {
                System.out.print("Enter student name: ");
                String name = sc.nextLine();
                System.out.print("Enter student ID: ");
                String id = sc.nextLine();
                records.add(new Student(name, id));
                System.out.println("Student added successfully!");
            } else if (choice == 2) {
                System.out.println("---- Student List ----");
                int i = 1;
                for (Student s : records) {
                    System.out.println(i + ". " + s.name + " (ID: " + s.id + ")");
                    i++;
                }
            } else {
                System.out.println("Thank you!");
                break;
            }
        }
    }
}
```

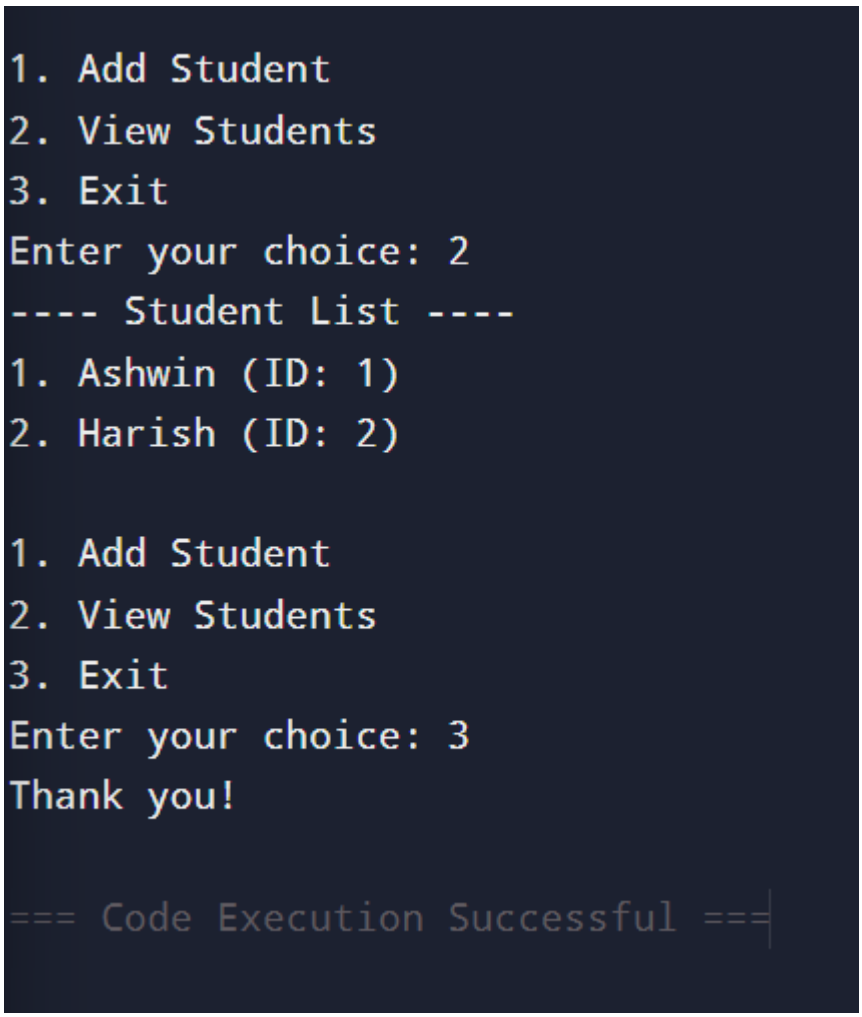
PROGRAM LOGIC:

1. Define a class Student with name and id attributes.
2. Maintain an ArrayList<Student> to store objects dynamically.
3. Menu allows adding or viewing student records.
4. Each added record persists until program termination.

OUTPUT:

```
1. Add Student
2. View Students
3. Exit
Enter your choice: 1
Enter student name: Ashwin
Enter student ID: 1
Student added successfully!

1. Add Student
2. View Students
3. Exit
Enter your choice: 1
Enter student name: Harish
Enter student ID: 2
Student added successfully!
```



CONCLUSION:

This exercise reinforces the use of **classes, constructors, and Array-Lists** for managing object collections, an essential concept for real-world data systems. Which has been done and executed successfully.

EXPERIMENT – 5

AIM:

Build a **canteen ordering system** that allows a user to order multiple food items and display the total payable amount.

PSEUDOCODE:

```
BEGIN
  CREATE Scanner sc
  INITIALIZE total = 0

  LOOP forever
    DISPLAY "--- Canteen Menu ---"
    DISPLAY "1. Sandwich ₹80"
    DISPLAY "2. Tea ₹20"
    DISPLAY "3. Juice ₹60"
    DISPLAY "4. Noodles ₹90"
    DISPLAY "5. Dessert ₹70"
    DISPLAY "6. Generate Bill"
    PROMPT for choice
    READ choice

    SWITCH(choice)
      CASE 1: ADD 80 to total; DISPLAY "Sandwich added!"; BREAK
      CASE 2: ADD 20 to total; DISPLAY "Tea added!"; BREAK
      CASE 3: ADD 60 to total; DISPLAY "Juice added!"; BREAK
      CASE 4: ADD 90 to total; DISPLAY "Noodles added!"; BREAK
      CASE 5: ADD 70 to total; DISPLAY "Dessert added!"; BREAK
```

```
            CASE 6:
               DISPLAY "---- BILL SUMMARY ----"
               DISPLAY "Total Payable: ₹" + total
               DISPLAY "Thank you! Visit again."
               EXIT program
            DEFAULT:
               DISPLAY "Invalid choice!"
         ENDSWITCH
      ENDLOOP
   END
```

PROGRAM:

```
   import java.util.*;

   public class Main {
      public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         int total = 0;

         while (true) {
            System.out.println("\n--- Canteen Menu ---");
            System.out.println("1. Sandwich ₹80");
            System.out.println("2. Tea ₹20");
            System.out.println("3. Juice ₹60");
            System.out.println("4. Noodles ₹90");
            System.out.println("5. Dessert ₹70");
            System.out.println("6. Generate Bill");
            System.out.print("Enter your choice: ");
```

```
            int ch = sc.nextInt();

            switch (ch) {
                case 1: total += 80; System.out.println("Sandwich added!"); break;
                case 2: total += 20; System.out.println("Tea added!"); break;
                case 3: total += 60; System.out.println("Juice added!"); break;
                case 4: total += 90; System.out.println("Noodles added!"); break;
                case 5: total += 70; System.out.println("Dessert added!"); break;
                case 6:
                    System.out.println("\n---- BILL SUMMARY ----");
                    System.out.println("Total Payable: ₹" + total);
                    System.out.println("Thank you! Visit again.");
                    return;
                default:
                    System.out.println("Invalid choice!");
            }
        }
    }
}
```

PROGRAM LOGIC:

1) Declare variable total to accumulate item prices.

2) Display menu and accept user selection in a loop.

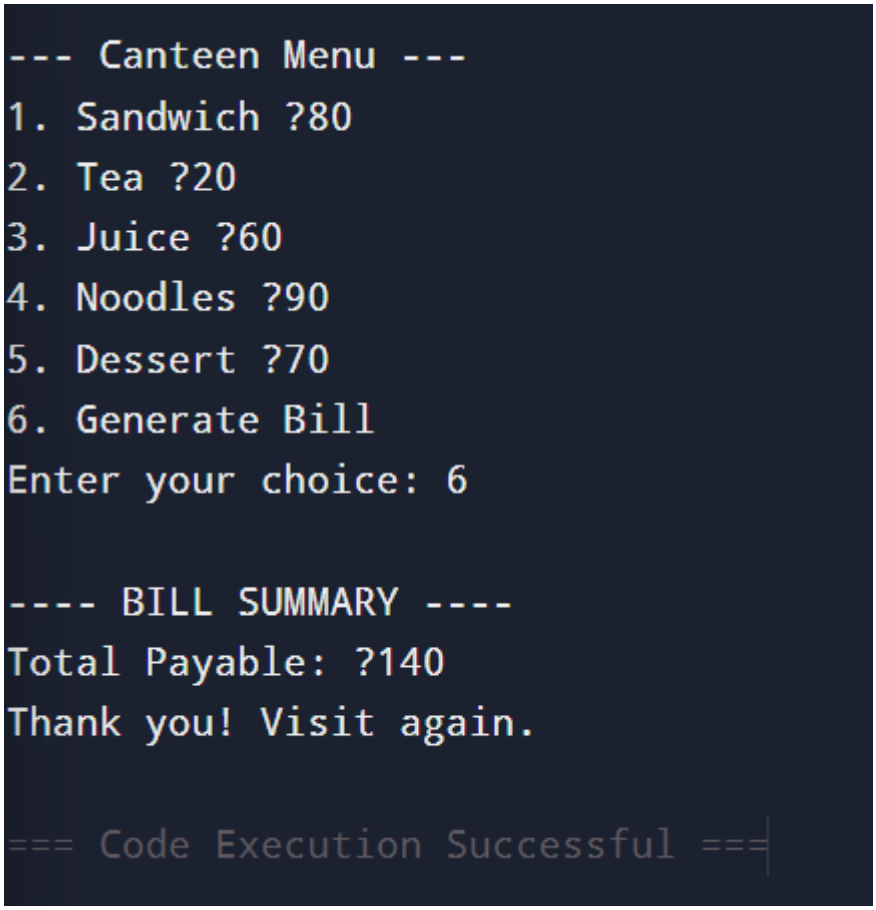
3) Add corresponding price to total for each choice.

4) Display final bill and exit when user selects billing option.

OUTPUT:

```
--- Canteen Menu ---
1. Sandwich ?80
2. Tea ?20
3. Juice ?60
4. Noodles ?90
5. Dessert ?70
6. Generate Bill
Enter your choice: 1
Sandwich added!

--- Canteen Menu ---
1. Sandwich ?80
2. Tea ?20
3. Juice ?60
4. Noodles ?90
5. Dessert ?70
6. Generate Bill
Enter your choice: 3
Juice added!
```



CONCLUSION:

This experiment reinforces decision structures and accumulative computation. It simulates a simple ordering system that strengthens logical control and practical application of loops and switch statements. Which has been done and executed successfully.