

## ASSIGNMENT – 3.3

**2303A51595**

**B-10**

### TASK-1

**Prompt :** Generate a Java program (only main method) to read Previous Units, Current Units, and Customer Type using Scanner. Calculate and display units consumed with simple comments.

#### CODE :

```
import java.util.Scanner;

public class Bill {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter previous units: ");

        int previousUnits = scanner.nextInt();

        System.out.print("Enter current units: ");

        int currentUnits = scanner.nextInt();

        if (currentUnits <= previousUnits) {

            System.out.println("Error: Current units must be greater than previous units.");

            scanner.close();

            return;

        }

        System.out.print("Enter type of consumer (Domestic / Commercial / Industrial): ");

        String consumerType = scanner.next();

        int unitsConsumed = currentUnits - previousUnits;

        System.out.println("Previous Units: " + previousUnits);

        System.out.println("Current Units: " + currentUnits);

        System.out.println("Type of Consumer: " + consumerType);

    }

}
```

```

        System.out.println("Units Consumed: " + unitsConsumed)

        scanner.close();
    }
}

```

## OUTPUT :

The screenshot shows the VS Code editor with the `Bill.java` file open. The code is as follows:

```

public class Bill {
    public static void main(String[] args) {
        System.out.println("Previous Units: " + previousUnits);
        System.out.println("Current Units: " + currentUnits);
        System.out.println("Type of Consumer: " + consumerType);
        System.out.println("Units Consumed: " + unitsConsumed);

        scanner.close();
    }
}

```

The terminal output shows the execution of the program:

```

C:\Users\deept\Downloads\OneDrive\Desktop\AI ASSISTED CODE>Bill.java
C:\Users\deept\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java Bill
Enter previous units: 4
Enter current units: 5
Enter type of consumer (Domestic / Commercial / Industrial): 23
Previous Units: 4
Current Units: 5
Type of Consumer: 23
Units Consumed: 1
C:\Users\deept\Downloads\OneDrive\Desktop\AI ASSISTED CODE>

```

## Analysis :

This task ensures that the electricity billing system captures accurate meter readings and customer information.

By calculating the difference between current and previous units, the program determines actual power consumption.

Implementing the logic directly in the main program keeps the execution straightforward and transparent.

This step is essential for building a reliable and realistic electricity bill generation system.

## TASK -2

**Prompt :** Generate a Java program that calculates Energy Charges using units consumed and customer type (Domestic, Commercial, Industrial) with optimized if-else conditions and clear output.

## CODE :

```
import java.util.Scanner;
```

```

public class EnergyBillCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter previous units: ");
        int previousUnits = scanner.nextInt();

        System.out.print("Enter current units: ");
        int currentUnits = scanner.nextInt();

        if (currentUnits <= previousUnits) {
            System.out.println("Error: Current units must be greater than previous units.");
            scanner.close();
            return;
        }

        System.out.print("Enter type of consumer (Domestic / Commercial / Industrial): ");
        String consumerType = scanner.next();

        int unitsConsumed = currentUnits - previousUnits;
        double chargePerUnit;

        switch (consumerType.toLowerCase()) {
            case "domestic":
                chargePerUnit = 5.0;
                break;
            case "commercial":
                chargePerUnit = 8.0;
                break;
            case "industrial":
                chargePerUnit = 10.0;
                break;
            default:
                System.out.println("Error: Invalid consumer type.");
                scanner.close();
                return;
        }

        double totalCharge = unitsConsumed * chargePerUnit;

        System.out.println("----- Energy Bill -----");
        System.out.println("Previous Units: " + previousUnits);
        System.out.println("Current Units: " + currentUnits);
        System.out.println("Type of Consumer: " + consumerType);
        System.out.println("Units Consumed: " + unitsConsumed);
        System.out.printf("Total Energy Charge: %.2f\n", totalCharge);

        scanner.close();
    }
}

```

```

    }
}

```

## OUTPUT :

The screenshot shows the VS Code editor with the file `EnergyBillCalculator.java` open. The code defines a `main` method that prompts the user for previous and current units, consumer type, and units consumed. It then calculates the total energy charge based on these inputs. The terminal output shows the program's execution with the following inputs: previous units: 4, current units: 5, consumer type: commercial, and units consumed: 1. The resulting total energy charge is 8.00.

```

C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java EnergyBillCalculator
Enter previous units: 4
Enter current units: 5
Enter type of consumer (Domestic / Commercial / Industrial): commercial
----- Energy Bill -----
Previous Units: 4
Current Units: 5
Type of Consumer: commercial
Units Consumed: 1
Total Energy Charge: 8.00

```

## Analysis :

In Task 2, the program is extended to calculate energy charges based on the number of units consumed and the type of consumer. Conditional logic is used to apply different charge rates for domestic, commercial, and industrial users. This ensures that billing follows realistic tariff rules. Optimizing the conditions improves code clarity and makes the billing logic easy to understand and modify.

## TASK – 3

**Prompt :** *Generate a Java program using user-defined methods to calculate **Energy Charges** and **Fixed Charges**, return values, and display the results with clear comments.*

## CODE :

```

import java.util.Scanner;

public class EnergyBillCalculator {

    // Method to calculate energy charges
    public static double calculateEnergyCharge(int unitsConsumed, String consumerType) {
        double chargePerUnit;

        switch (consumerType.toLowerCase()) {

```

```

        case "domestic":
            chargePerUnit = 5.0;
            break;
        case "commercial":
            chargePerUnit = 8.0;
            break;
        case "industrial":
            chargePerUnit = 10.0;
            break;
        default:
            throw new IllegalArgumentException("Invalid consumer type.");
    }
    return unitsConsumed * chargePerUnit;
}

```

// Method to calculate fixed charges

```

public static double calculateFixedCharge(String consumerType) {
    switch (consumerType.toLowerCase()) {
        case "domestic":
            return 50.0;
        case "commercial":
            return 100.0;
        case "industrial":
            return 150.0;
        default:
            throw new IllegalArgumentException("Invalid consumer type.");
    }
}

```

```

public static void main(String[] args) {

```

```

    Scanner scanner = new Scanner(System.in);

```

```

    System.out.print("Enter previous units: ");
    int previousUnits = scanner.nextInt();

```

```

    System.out.print("Enter current units: ");
    int currentUnits = scanner.nextInt();

```

```

    if (currentUnits <= previousUnits) {
        System.out.println("Error: Current units must be greater than previous units.");
        scanner.close();
        return;
    }

```

```

    System.out.print("Enter type of consumer (Domestic / Commercial / Industrial): ");
    String consumerType = scanner.next();

```

```

    int unitsConsumed = currentUnits - previousUnits;

```

```

    double energyCharge = calculateEnergyCharge(unitsConsumed, consumerType);
    double fixedCharge = calculateFixedCharge(consumerType);

```

```
double totalCharge = energyCharge + fixedCharge;

System.out.println("----- Energy Bill -----");
System.out.println("Units Consumed: " + unitsConsumed);
System.out.printf("Energy Charge: %.2f\n", energyCharge);
System.out.printf("Fixed Charge: %.2f\n", fixedCharge);
System.out.printf("Total Charge: %.2f\n", totalCharge);

scanner.close();
}
}
```

## OUTPUT :

The screenshot shows a code editor with the following Java code in `EnergyBillCalculator.java`:

```
97 public class EnergyBillCalculator {
98     // Method to calculate energy charges
99     public static double calculateEnergyCharge(int unitsConsumed, String consumerType) {
100         double chargePerUnit;
101
102         switch (consumerType.toLowerCase()) {
103             case "domestic":
104                 chargePerUnit = 5.0;
105                 break;
106             case "commercial":
107                 chargePerUnit = 8.0;
108                 break;
109         }
110     }
111 }
```

The terminal output shows the program execution:

```
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>javac EnergyBillCalculator.java
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java EnergyBillCalculator
Enter previous units: 4
Enter current units: 5
Enter type of consumer (Domestic / Commercial / Industrial): domestic
----- Energy Bill -----
Units Consumed: 1
Energy Charge: 5.00
Fixed Charge: 50.00
Total Charge: 55.00
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>
```

## Analysis :

Task 3 introduces modular programming using user-defined methods. Separate methods are used to calculate energy charges and fixed charges.

This makes the program reusable and easier to maintain. Modular design improves code structure and readability.

## TASK- 4

**Prompt :** Write a Java electricity billing program to calculate *Fixed Charges*, *Customer Charges*, and *Electricity Duty* (as a percentage of *Energy Charges*), print each charge separately, and improve billing accuracy with clear output.

## CODE :

```

import java.util.Scanner;
public class EnergyBillCalculator {

    // Method to calculate energy charges
    public static double calculateEnergyCharge(int unitsConsumed, String consumerType) {
        double chargePerUnit;

        switch (consumerType.toLowerCase()) {
            case "domestic":
                chargePerUnit = 5.0;
                break;
            case "commercial":
                chargePerUnit = 8.0;
                break;
            case "industrial":
                chargePerUnit = 10.0;
                break;
            default:
                throw new IllegalArgumentException("Invalid consumer type.");
        }
        return unitsConsumed * chargePerUnit;
    }

    // Method to calculate fixed charges
    public static double calculateFixedCharge(String consumerType) {
        switch (consumerType.toLowerCase()) {
            case "domestic":
                return 50.0;
            case "commercial":
                return 100.0;
            case "industrial":
                return 150.0;
            default:
                throw new IllegalArgumentException("Invalid consumer type.");
        }
    }

    // Method to calculate customer charges
    public static double calculateCustomerCharge(String consumerType) {
        switch (consumerType.toLowerCase()) {
            case "domestic":
                return 20.0;
            case "commercial":
                return 40.0;
            case "industrial":
                return 60.0;
            default:
                throw new IllegalArgumentException("Invalid consumer type.");
        }
    }

    // Method to calculate electricity duty

```

```

public static double calculateElectricityDuty(double energyCharge) {
    return energyCharge * 0.05; // 5% of energy charges
}

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter previous units: ");
    int previousUnits = scanner.nextInt();

    System.out.print("Enter current units: ");
    int currentUnits = scanner.nextInt();

    if (currentUnits <= previousUnits) {
        System.out.println("Error: Current units must be greater than previous units.");
        scanner.close();
        return;
    }

    System.out.print("Enter type of consumer (Domestic / Commercial / Industrial): ");
    String consumerType = scanner.next();

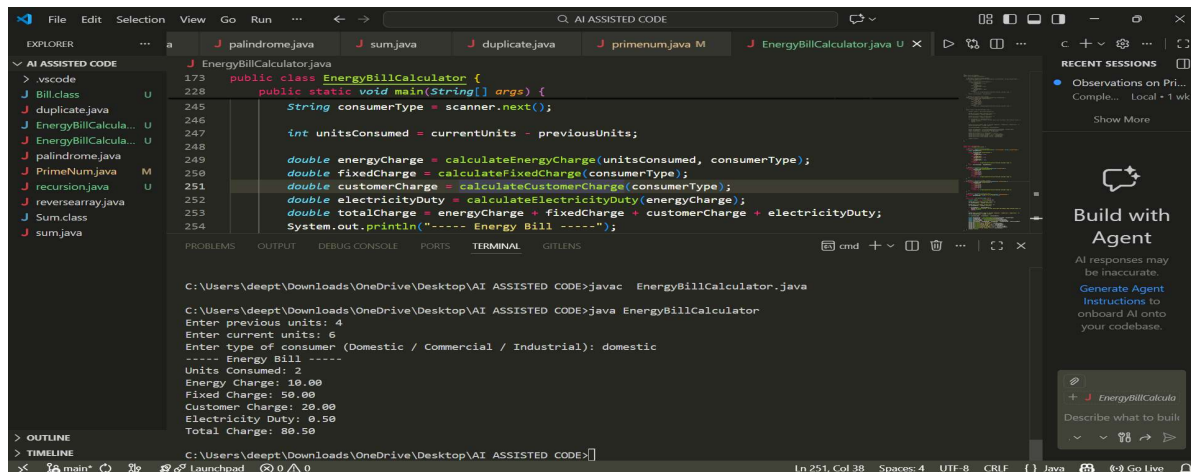
    int unitsConsumed = currentUnits - previousUnits;

    double energyCharge = calculateEnergyCharge(unitsConsumed, consumerType);
    double fixedCharge = calculateFixedCharge(consumerType);
    double customerCharge = calculateCustomerCharge(consumerType);
    double electricityDuty = calculateElectricityDuty(energyCharge);
    double totalCharge = energyCharge + fixedCharge + customerCharge + electricityDuty;
    System.out.println("----- Energy Bill -----");
    System.out.println("Units Consumed: " + unitsConsumed);
    System.out.printf("Energy Charge: %.2f\n", energyCharge);
    System.out.printf("Fixed Charge: %.2f\n", fixedCharge);
    System.out.printf("Customer Charge: %.2f\n", customerCharge);
    System.out.printf("Electricity Duty: %.2f\n", electricityDuty);
    System.out.printf("Total Charge: %.2f\n", totalCharge);
    scanner.close();
}
}

```

**OUTPUT :**





## Analysis :

This task extends billing by adding additional charges like fixed charges, customer charges, and electricity duty.

Electricity duty is calculated as a percentage of energy charges.

Printing individual charges helps verify calculation accuracy.

This step makes the bill more realistic and detailed.

## TASK – 5

### Prompt :

*Generate a Java electricity billing program that calculates total bill (EC + FC + CC + ED), displays a neatly formatted final bill with all charge components, and includes brief comments on accuracy, readability, and real-world applicability.*

### CODE :

```

import java.util.Scanner;
public class EnergyBillCalculator {
    // Method to calculate energy charges
    public static double calculateEnergyCharge(int unitsConsumed, String consumerType) {
        double chargePerUnit;

        switch (consumerType.toLowerCase()) {
            case "domestic":
                chargePerUnit = 5.0;
                break;
            case "commercial":
  
```

```

        chargePerUnit = 8.0;
        break;
    case "industrial":
        chargePerUnit = 10.0;
        break;
    default:
        throw new IllegalArgumentException("Invalid consumer type.");
    }
    return unitsConsumed * chargePerUnit;
}

```

```

// Method to calculate fixed charges
public static double calculateFixedCharge(String consumerType) {
    switch (consumerType.toLowerCase()) {
        case "domestic":
            return 50.0;
        case "commercial":
            return 100.0;
        case "industrial":
            return 150.0;
        default:
            throw new IllegalArgumentException("Invalid consumer type.");
    }
}

```

```

// Method to calculate customer charges
public static double calculateCustomerCharge(String consumerType) {
    switch (consumerType.toLowerCase()) {
        case "domestic":
            return 20.0;
        case "commercial":
            return 40.0;
        case "industrial":
            return 60.0;
        default:
            throw new IllegalArgumentException("Invalid consumer type.");
    }
}

```

```

// Method to calculate electricity duty
public static double calculateElectricityDuty(double energyCharge) {
    return energyCharge * 0.05; // 5% of energy charges
}

```

```

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter previous units: ");
    int previousUnits = scanner.nextInt();

    System.out.print("Enter current units: ");
}

```

```

int currentUnits = scanner.nextInt();

if (currentUnits <= previousUnits) {
    System.out.println("Error: Current units must be greater than previous units.");
    scanner.close();
    return;
}

System.out.print("Enter type of consumer (Domestic / Commercial / Industrial): ");
String consumerType = scanner.next();

int unitsConsumed = currentUnits - previousUnits;

double energyCharge = calculateEnergyCharge(unitsConsumed, consumerType);
double fixedCharge = calculateFixedCharge(consumerType);
double customerCharge = calculateCustomerCharge(consumerType);
double electricityDuty = calculateElectricityDuty(energyCharge);
double totalCharge = energyCharge + fixedCharge + customerCharge + electricityDuty;
System.out.println("----- Energy Bill -----");
System.out.println("Units Consumed: " + unitsConsumed);
System.out.printf("Energy Charge: %.2f\n", energyCharge);
System.out.printf("Fixed Charge: %.2f\n", fixedCharge);
System.out.printf("Customer Charge: %.2f\n", customerCharge);
System.out.printf("Electricity Duty: %.2f\n", electricityDuty);
System.out.printf("Total Charge: %.2f\n", totalCharge);
scanner.close();
}

```

## OUTPUT :

```

C:\Users\deept\Downloads\OneDrive\Desktop\AI_ASSISTED_CODE>javac EnergyBillCalculator.java

C:\Users\deept\Downloads\OneDrive\Desktop\AI_ASSISTED_CODE>java EnergyBillCalculator
Enter previous units: 4
Enter current units: 5
Enter type of consumer (Domestic / Commercial / Industrial): commercial
----- Energy Bill -----
Units Consumed: 1
Energy Charge: 8.00
Fixed Charge: 100.00
Customer Charge: 40.00
Electricity Duty: 0.40
Total Charge: 148.40

```

## Analysis :

Task 5 generates the final electricity bill by combining all charge components.

The total bill amount is calculated by adding EC, FC, CC, and ED.

The output is displayed in a neat, bill-like format for clarity.

This task demonstrates a complete, real-world electricity billing application.