

## 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 a Java code editor interface with an AI-assisted feature. The code in Bill.java is:

```

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 window shows the execution of the program:

```

C:\Users\deepthi\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

```

A sidebar on the right says "Build with Agent" and "AI responses may be inaccurate".

## 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 a Java code editor in VS Code with the following 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;
        }
    }
}
```

The terminal below shows the execution of the code:

```
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 -----
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 the VS Code interface with the AI ASSISTED CODE extension active. The terminal window displays the execution of the Java program:

```

C:\Users\deepthi\Downloads\OneDrive\Desktop\AI ASSISTED CODE>javac EnergyBillCalculator.java
C:\Users\deepthi\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

```

## 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 :**

```

File Edit Selection View Go Run ... < > Q AI ASSISTED CODE
EXPLORER ... a J palindrome.java J sum.java J duplicate.java J primeNum.java M J EnergyBillCalculator.java U ...
RECENT SESSIONS
Observations on Pri... Complete... Local - 1 wk
Show More
AI ASSISTED CODE
J EnergyBillCalculator.java
173 public class EnergyBillCalculator {
174     public static void main(String[] args) {
175         Scanner scanner = new Scanner(System.in);
176         String consumerType = scanner.nextLine();
177         int previousUnits = scanner.nextInt();
178         int currentUnits = scanner.nextInt();
179         int unitsConsumed = currentUnits - previousUnits;
180         double energyCharge = calculateEnergyCharge(unitsConsumed, consumerType);
181         double fixedCharge = calculateFixedCharge(consumerType);
182         double customerCharge = calculateCustomerCharge(consumerType);
183         double electricityDuty = calculateElectricityDuty(energyCharge);
184         double totalCharge = energyCharge + fixedCharge + customerCharge + electricityDuty;
185         System.out.println("----- Energy Bill -----");
186         System.out.println("Unit Consumed: " + unitsConsumed);
187         System.out.println("Energy Charge: " + energyCharge);
188         System.out.println("Fixed Charge: " + fixedCharge);
189         System.out.println("Customer Charge: " + customerCharge);
190         System.out.println("Electricity Duty: " + electricityDuty);
191         System.out.println("Total Charge: " + totalCharge);
192     }
193     private static double calculateEnergyCharge(int unitsConsumed, String consumerType) {
194         double chargePerUnit;
195         switch (consumerType.toLowerCase()) {
196             case "domestic":
197                 chargePerUnit = 5.0;
198                 break;
199             case "commercial":
200                 chargePerUnit = 10.0;
201                 break;
202             default:
203                 chargePerUnit = 0.0;
204         }
205         return chargePerUnit * unitsConsumed;
206     }
207     private static double calculateFixedCharge(String consumerType) {
208         double fixedCharge;
209         if (consumerType.equals("domestic")) {
210             fixedCharge = 10.0;
211         } else if (consumerType.equals("commercial")) {
212             fixedCharge = 20.0;
213         } else {
214             fixedCharge = 0.0;
215         }
216         return fixedCharge;
217     }
218     private static double calculateCustomerCharge(String consumerType) {
219         double customerCharge;
220         if (consumerType.equals("domestic")) {
221             customerCharge = 10.0;
222         } else if (consumerType.equals("commercial")) {
223             customerCharge = 20.0;
224         } else {
225             customerCharge = 0.0;
226         }
227         return customerCharge;
228     }
229     private static double calculateElectricityDuty(double energyCharge) {
230         double electricityDuty;
231         electricityDuty = energyCharge * 0.05;
232         return electricityDuty;
233     }
234 }

```

C:\Users\deepthi\Downloads\OneDrive\Desktop\AI ASSISTED CODE>javac EnergyBillCalculator.java  
C:\Users\deepthi\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java EnergyBillCalculator  
Enter previous units: 4  
Enter current units: 6  
Enter type of consumer (Domestic / Commercial / Industrial): domestic  
----- Energy Bill -----  
Unit Consumed: 2  
Energy Charge: 10.00  
Fixed Charge: 10.00  
Customer Charge: 20.00  
Electricity Duty: 0.50  
Total Charge: 40.50

## 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 = 10.0;
                break;
            default:
                chargePerUnit = 0.0;
        }
        return chargePerUnit * unitsConsumed;
    }

    private static double calculateFixedCharge(String consumerType) {
        double fixedCharge;
        if (consumerType.equals("domestic")) {
            fixedCharge = 10.0;
        } else if (consumerType.equals("commercial")) {
            fixedCharge = 20.0;
        } else {
            fixedCharge = 0.0;
        }
        return fixedCharge;
    }

    private static double calculateCustomerCharge(String consumerType) {
        double customerCharge;
        if (consumerType.equals("domestic")) {
            customerCharge = 10.0;
        } else if (consumerType.equals("commercial")) {
            customerCharge = 20.0;
        } else {
            customerCharge = 0.0;
        }
        return customerCharge;
    }

    private static double calculateElectricityDuty(double energyCharge) {
        double electricityDuty;
        electricityDuty = energyCharge * 0.05;
        return electricityDuty;
    }
}

```

```

        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 :

The screenshot shows the AI ASSISTED CODE interface in VS Code. The Explorer sidebar lists several Java files. The terminal window displays the execution of the `EnergyBillCalculator.java` file, showing the user interaction and the calculated bill details.

```

File Edit Selection View Go Run ... ← → Q AI ASSISTED CODE
EXPLORER ... J palindrome.java J sumjava J duplicate.java J primenum.java M J EnergyBillCalculator.java U ...
RECENT SESSIONS ...
Observations on Pri...
Comple... Local • 1 wk
Show More
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

C:\Users\deepthi\Downloads\OneDrive\Desktop\AI ASSISTED CODE>javac EnergyBillCalculator.java
C:\Users\deepthi\Downloads\OneDrive\Desktop\AI ASSISTED CODE>
C:\Users\deepthi\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.