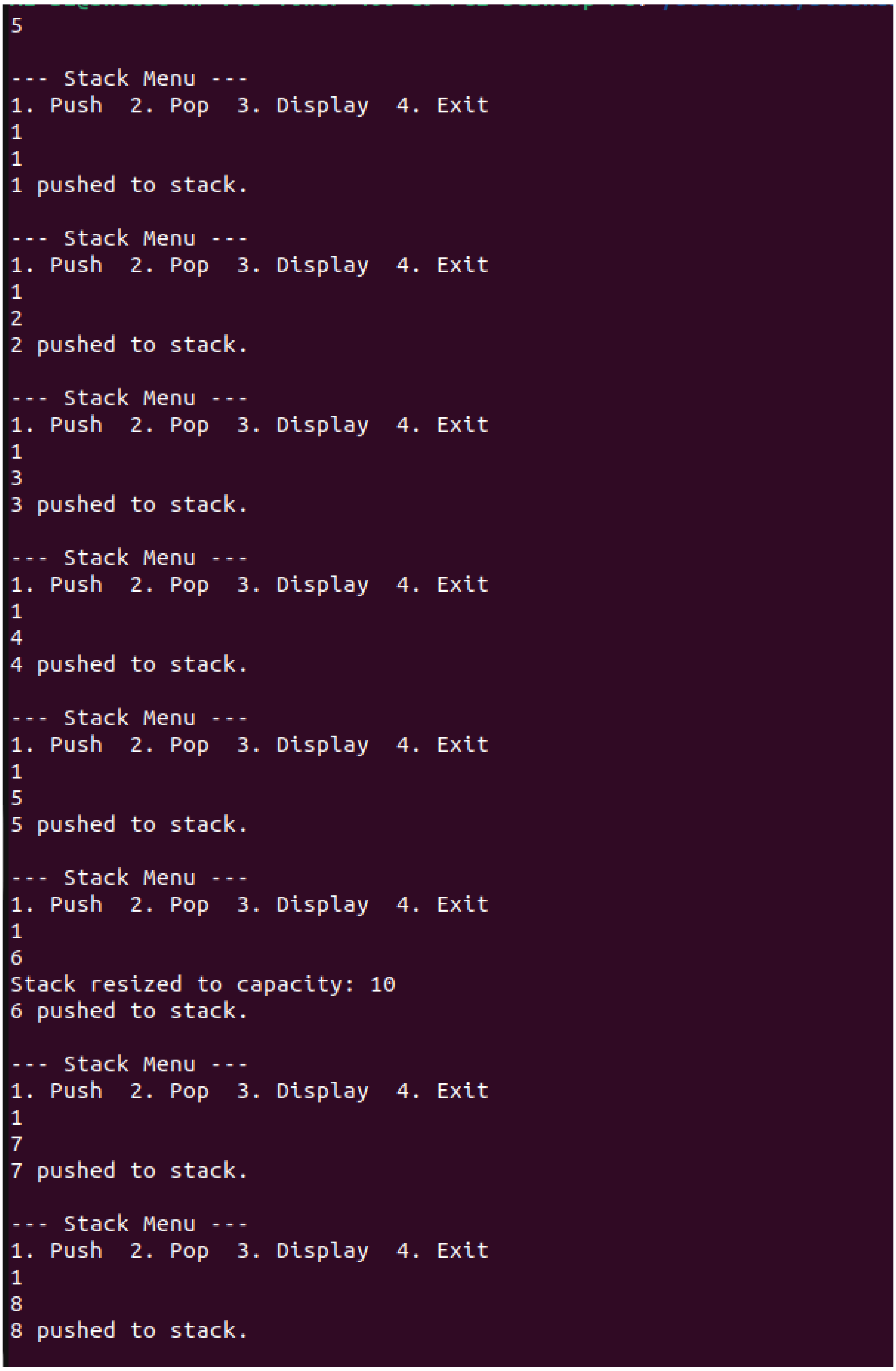
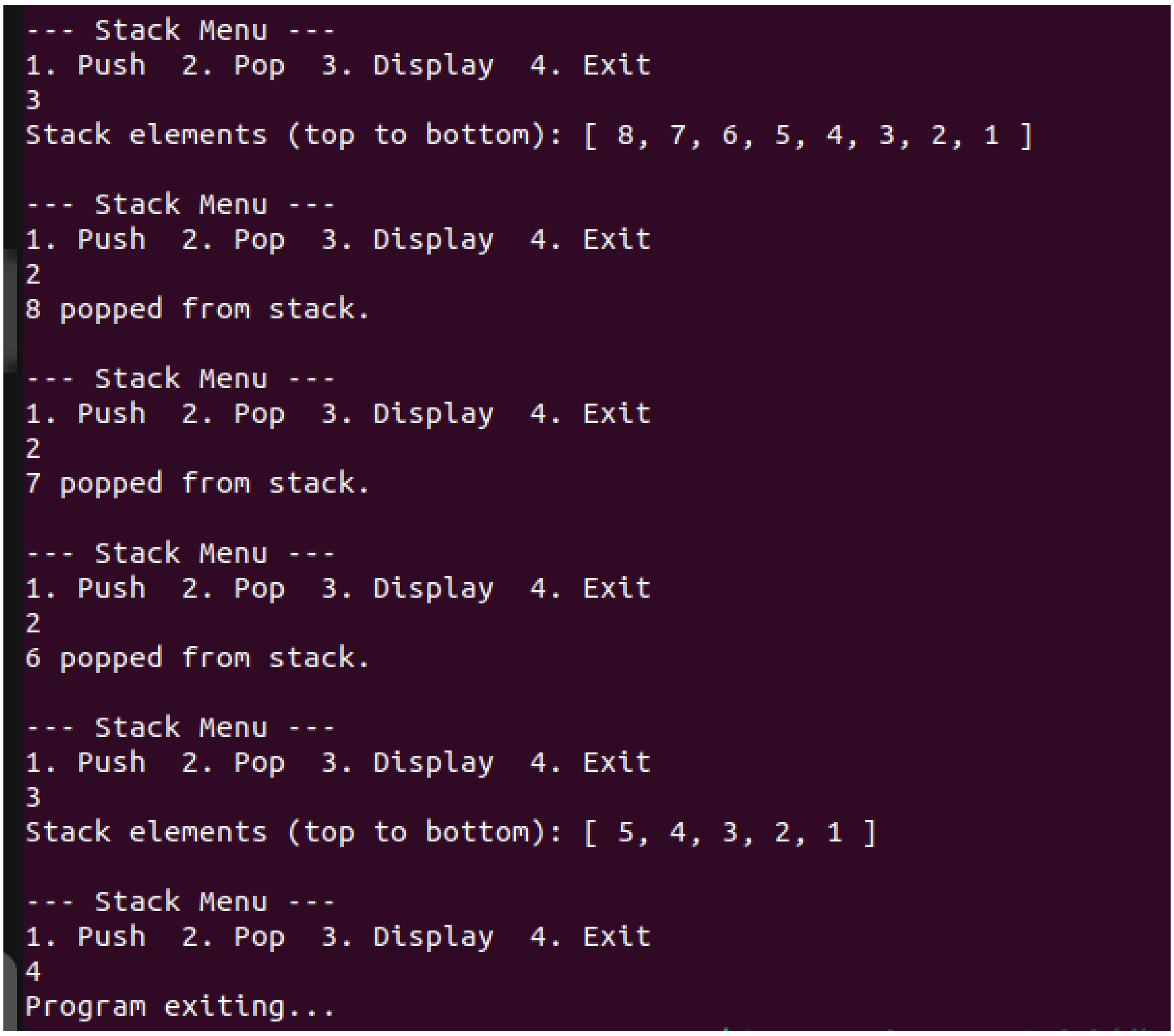
```java
import java.io.FileWriter;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.util.Scanner;

public class EBConsumer {
    public static void main(String[] args) throws Exception {
        // Setup log file and dual output
        FileWriter fw = new FileWriter("eb_log.txt", true); // Append mode
        PrintWriter logWriter = new PrintWriter(fw, true);
        PrintStream originalOut = System.out;

        // Redirect output to both console and file
        PrintStream logOut = new PrintStream(new OutputStream() {
            public void write(int b) {
                originalOut.write(b);
                logWriter.write(b);
            }
        }, true);
        System.setOut(logOut);

        // Run consumer logic
        Consumer c1 = new Consumer(logWriter);
        c1.inputs();
        c1.calculate();

        logWriter.close();
        fw.close();
    }
}

class Consumer {
    int c_no;
    String c_name;
    double prev_reading;
    double cur_reading;
    String c_type;
    double units;
    double amount = 0;

    PrintWriter logWriter;

    public Consumer(PrintWriter writer) {
        this.logWriter = writer;
    }

    public void inputs() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Consumer Number: ");
        if (sc.hasNextInt()) {
            c_no = sc.nextInt();
            sc.nextLine();
            logWriter.println("INPUT Consumer Number: " + c_no);
        } else {
            System.out.println("Consumer Number Should Be an Integer");
            sc.nextLine();
        }

        System.out.print("Enter Consumer Name: ");
        c_name = sc.nextLine();
        logWriter.println("INPUT Consumer Name: " + c_name);
        if (c_name.trim().isEmpty()) {
            System.out.println("Consumer Name Should Not Be Empty");
        }
        System.out.print("Enter Previous Reading: ");
       if (sc.hasNextDouble()) {
            prev_reading = sc.nextDouble();
            sc.nextLine();
            logWriter.println("INPUT Previous Reading: " + prev_reading);
            if (prev_reading < 0) {
                System.out.println("Reading Should Be a Positive Real Number");
            }
        } else {
            System.out.println("Reading Should Be a Positive Real Number");
            sc.nextLine();
        }

        System.out.print("Enter Current Reading: ");
        if (sc.hasNextDouble()) {
            cur_reading = sc.nextDouble();
            sc.nextLine();
            logWriter.println("INPUT Current Reading: " + cur_reading);
            if (cur_reading < 0) {
                System.out.println("Reading Should Be a Positive Real Number");
            } else if (cur_reading < prev_reading) {
                System.out.println("Current Reading cannot be less than Previous
Reading");
            }
        } else {
            System.out.println("Reading Should Be a Positive Real Number");
            sc.nextLine();
        }

        System.out.print("Enter Type (commercial/domestic): ");
        c_type = sc.nextLine();
        logWriter.println("INPUT Type: " + c_type);
        if (!(c_type.equalsIgnoreCase("commercial") ||
c_type.equalsIgnoreCase("domestic"))) {
            System.out.println("Consumer Type Must Be 'commercial' or 'domestic'");
        }

        units = cur_reading - prev_reading;

        sc.close();
    }

    public void calculate() {
        if (c_type.equalsIgnoreCase("domestic")) {
            if (units <= 100) {
                amount = units * 1;
            } else if (units <= 200) {
                amount = 100 * 1 + (units - 100) * 2.5;
            } else if (units <= 500) {
                amount = 100 * 1 + 100 * 2.5 + (units - 200) * 4;
            } else {
                amount = 100 * 1 + 100 * 2.5 + 300 * 4 + (units - 500) * 6;
            }
        } else if (c_type.equalsIgnoreCase("commercial")) {
            if (units <= 100) {
                amount = units * 2;
            } else if (units <= 200) {
                amount = 100 * 2 + (units - 100) * 4.5;
            } else if (units <= 500) {
                amount = 100 * 2 + 100 * 4.5 + (units - 200) * 6;
            } else {
                amount = 100 * 2 + 100 * 4.5 + 300 * 6 + (units - 500) * 7;
            }
        } else {
            System.out.println("Invalid Type");
            return;
        }

        // Output bill
        System.out.println("\n--- Electricity Bill ---");
        System.out.println("Consumer Number: " + c_no);
        System.out.println("Consumer Name: " + c_name);
        System.out.println("Consumer Type: " + c_type);
        System.out.println("Units Consumed: " + units);
        System.out.println("Amount to Pay: " + amount);

        // Also log it explicitly (redundant, since System.out is already logging to file)
        logWriter.println("OUTPUT Units: " + units);
        logWriter.println("OUTPUT Amount: " + amount);
    }
}
```

```
Enter Consumer Number: INPUT Consumer Number: 454454
Enter Consumer Name: INPUT Consumer Name: Siddharth
Enter Previous Reading: INPUT Previous Reading: 4569.0
Enter Current Reading: INPUT Current Reading: 4862.0
Enter Type (commercial/domestic): INPUT Type: domestic

--- Electricity Bill ---
Consumer Number: 454454
Consumer Name: Siddharth
Consumer Type: domestic
Units Consumed: 293.0
Amount to Pay: 722.0
OUTPUT Units: 293.0
OUTPUT Amount: 722.0

Enter Consumer Number: INPUT Consumer Number: 755486
Enter Consumer Name: INPUT Consumer Name: 10025
Enter Previous Reading: INPUT Previous Reading: 12589.0
Enter Current Reading: INPUT Current Reading: 14859.0
Enter Type (commercial/domestic): INPUT Type: comercial
Consumer Type Must Be 'commercial' or 'domestic'
Invalid Type

Enter Consumer Number: INPUT Consumer Number: 755486
Enter Consumer Name: INPUT Consumer Name: IBM
Enter Previous Reading: INPUT Previous Reading: 10025.0
Enter Current Reading: INPUT Current Reading: 12589.0
Enter Type (commercial/domestic): INPUT Type: commercial

--- Electricity Bill ---
Consumer Number: 755486
Consumer Name: IBM
Consumer Type: commercial
Units Consumed: 2564.0
Amount to Pay: 16898.0
OUTPUT Units: 2564.0
OUTPUT Amount: 16898.0

Enter Consumer Number: INPUT Consumer Number: 456923
Enter Consumer Name: INPUT Consumer Name: ABC TEXTILES
Enter Previous Reading: INPUT Previous Reading: 10025.0
Enter Current Reading: INPUT Current Reading: 9096.0
Current Reading cannot be less than Previous Reading

Enter Consumer Number: Consumer Number Should Be an Integer
Enter Consumer Name: INPUT Consumer Name: 4454
Enter Previous Reading: INPUT Previous Reading: 5421.0
Enter Current Reading: Reading Should Be a Positive Real Number
Enter Type (commercial/domestic): INPUT Type: commmercial
Consumer Type Must Be 'commercial' or 'domestic'
Invalid Type
```



```
5

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
1
1 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
2
2 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
3
3 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
4
4 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
5
5 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
6
Stack resized to capacity: 10
6 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
7
7 pushed to stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
1
8
8 pushed to stack.
```



```
--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
3
Stack elements (top to bottom): [ 8, 7, 6, 5, 4, 3, 2, 1 ]

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
2
8 popped from stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
2
7 popped from stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
2
6 popped from stack.

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
3
Stack elements (top to bottom): [ 5, 4, 3, 2, 1 ]

--- Stack Menu ---
1. Push  2. Pop  3. Display  4. Exit
4
Program exiting...
```