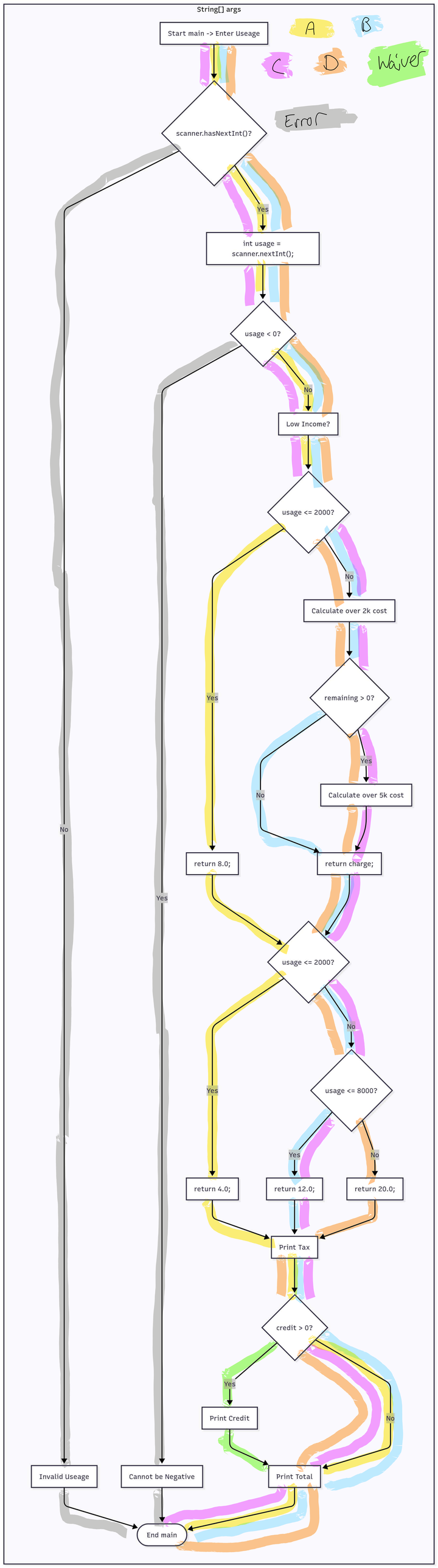
String[] args
Start main -> Enter Useage
A
B
C
D
Waiver
Error
scanner.hasNextInt()?
Yes
No
int usage =
scanner.nextInt();
usage < 0?
No
Yes
Low Income?
usage <= 2000?
Yes
No
Calculate over 2k cost
remaining > 0?
Yes
No
Calculate over 5k cost
return 8.0;
return charge;
usage <= 2000?
Yes
No
usage <= 8000?
Yes
No
return 4.0;
return 12.0;
return 20.0;
Print Tax
credit > 0?
Yes
No
Print Credit
Print Total
Invalid Useage
Cannot be Negative
End main

I could not test the paths that would require usage to be two different values simultaneously, for obvious reasons. I did not have full coverage for the paths with waivers, or input checking that ChatGPT included, and so will add a second table with them before my original table. I over-tested for some of the categories by doing border tests, EX: the over 2k, and under 5k hit the same nodes and paths. The graph is greatly simplified because trying to include all the lines would be a nightmare, and you wont be able to see line numbers, but I tried to statements from the code that should be relatively easy to identify.

New Test Cases	
Within 2k limit, W/ Waiver	
Input	2000
	No Waiver
Expected Output	8
	Base + delivery - waiver
Actual Output	8
	Base + delivery - waiver
Pass/Fail	Pass
Path	A W/ Waiver
Over 2k limit, W/ Waiver	
Input	2002
	No Waiver
Expected Output	16.02
	Base + delivery - waiver
Actual Output	16.02
	Base + delivery - waiver
Pass/Fail	Pass
Path	B W/ Waiver
Improper usage entered	
Input	b
Expected	Improper Usage
Actual	Improper Usage
Pass/Fail	Pass
Improper usage entered	
Input	-5
Expected	Improper Usage
Actual	Improper Usage
Pass/Fail	Pass

Within 2k limit, no waiver	
Input	2000
	No Waiver
Expected Output	12.2
	Base + delivery + tax
Actual Output	12.2
	Base + delivery + tax
Pass/Fail	Pass
Path	A
Over 2k limit, no waiver	
Input	2002
	No Waiver
Expected Output	20.21
	Base + delivery + tax
Actual Output	20.21
	Base + delivery + tax
Pass/Fail	Pass
Path	B
Within 5k limit, no waiver	
Input	5000
	No Waiver
Expected Output	32.5
	Base + delivery + tax
Actual Output	32.5
	Base + delivery + tax
Pass/Fail	Pass
Path	
Over 5k limit, no waiver	
Input	5002
	No Waiver
Expected Output	32.51
	Base + delivery + tax
Actual Output	32.51
	Base + delivery + tax
Pass/Fail	Pass
Path	C

Over 8k limit, no waiver	
Input	8002
	No Waiver
Expected Output	62.04
	Base + delivery + tax
Actual Output	62.04
	Base + delivery + tax
Pass/Fail	Pass
Path	D
Over 5k limit, W/ waiver	
Input	5002
	Waiver
Expected Output	28.01
	Base + delivery + tax
Actual Output	28.01
	Base + delivery + tax
Pass/Fail	Pass
Path	C W/ Waiver
Over 8k limit, W/ waiver	
Input	8002
	Waiver
Expected Output	61.01
	Base + delivery + tax
Actual Output	61.01
	Base + delivery + tax
Pass/Fail	Pass
Path	D W/ Waiver

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

public class proj04 {
   public static void main(String[] args) {
      Locale.setDefault(Locale.US);
      try (Scanner scanner = new Scanner(System.in)) {
         System.out.print("Enter monthly usage in gallons: ");
         if (!scanner.hasNextInt()) {
            System.out.println("Invalid usage. Please provide a non-negative whole number of gallons.");
```

```
    return;
}

int usage = scanner.nextInt();
if (usage < 0) {
    System.out.println("Invalid usage. Usage cannot be negative.");
    return;
}

System.out.print("Is this a low-income household? (y/n): ");
boolean lowIncome = false;
if (scanner.hasNext()) {
    String response = scanner.next().trim().toLowerCase(Locale.US);
    lowIncome = response.startsWith("y");
}

double waterCharge = calculateWaterCharge(usage);
double surcharge = calculateSurcharge(usage);
double tax = lowIncome ? 0.0 : waterCharge * 0.025;
double credit = (lowIncome && usage <= 8000) ? 4.0 : 0.0;
double totalDue = Math.max(0.0, waterCharge + surcharge + tax - credit);

System.out.println();
System.out.printf("Usage:%18d gallons%n", usage);
System.out.printf("Water charge:%11.2f%n", waterCharge);
System.out.printf("Surcharge:%12.2f%n", surcharge);
System.out.printf("Tax:%18.2f%n", tax);
if (credit > 0) {
    System.out.printf("Low-income credit:%4.2f%n", credit);
}
```

```
        System.out.printf("Total due:%13.2f%n", totalDue);
    }
}

private static double calculateWaterCharge(int usage) {
    if (usage <= 2000) {
        return 8.0;
    }

    double charge = 8.0;
    int remaining = usage - 2000;

    int midTierGallons = Math.min(remaining, 3000);
    charge += midTierGallons * 0.004;
    remaining -= midTierGallons;

    if (remaining > 0) {
        charge += remaining * 0.007;
    }

    return charge;
}

private static double calculateSurcharge(int usage) {
    if (usage <= 2000) {
        return 4.0;
    }
    if (usage <= 8000) {
        return 12.0;
    }
```

```
        return 20.0;
    }
}
```