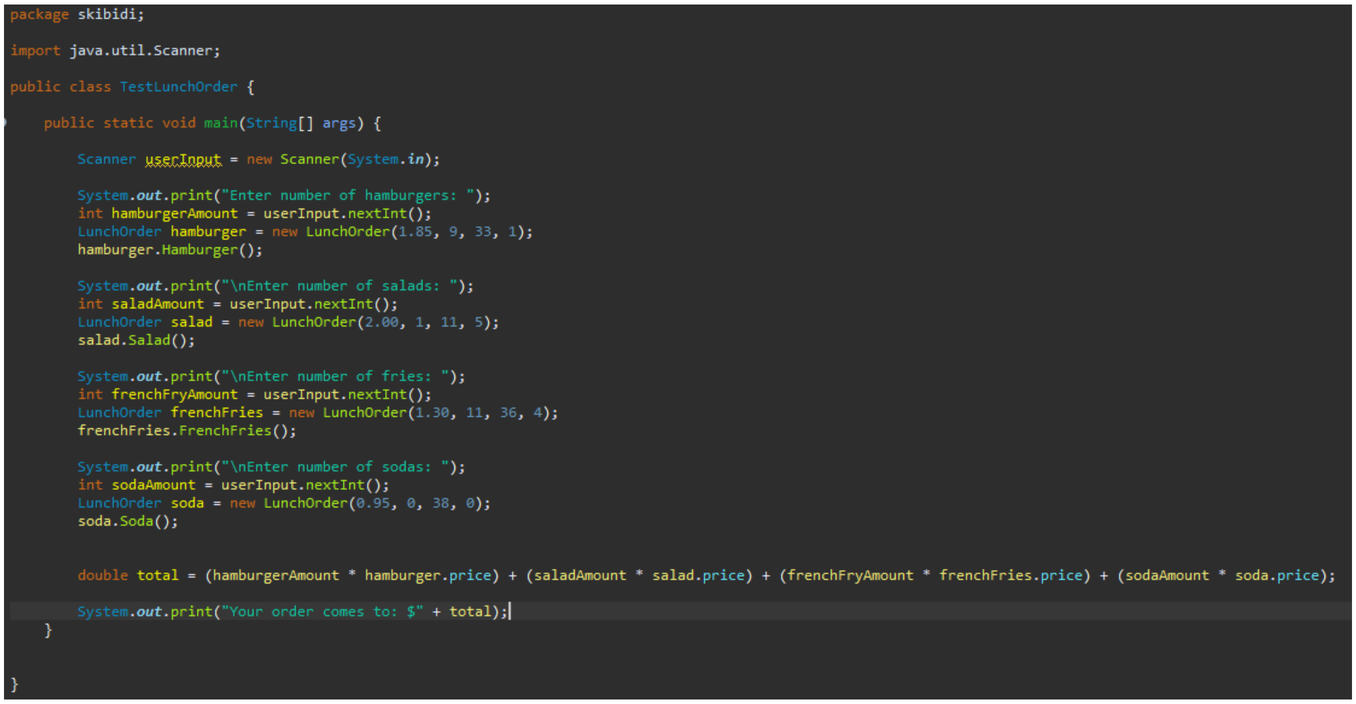
DigitExtractor Mastery ReflectionLog

Foreword: Screenshots were taken on both my home and school computer, hence the difference between the package in some of the screenshots.

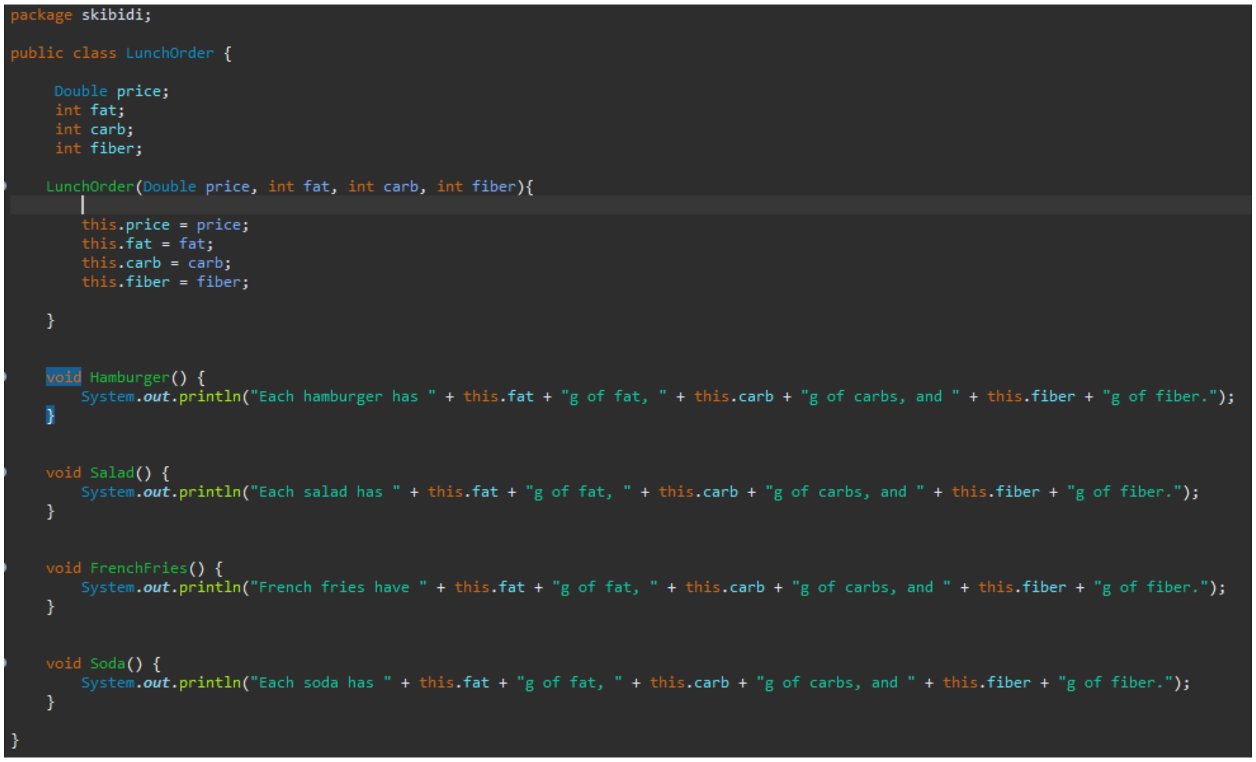
Below is the first rendition of the TestLunchOrder class:



```
package skibidi;

import java.util.Scanner;

public class TestLunchOrder {

    public static void main(String[] args) {

        Scanner userInput = new Scanner(System.in);

        System.out.print("Enter number of hamburgers: ");
        int hamburgerAmount = userInput.nextInt();
        LunchOrder hamburger = new LunchOrder(1.85, 9, 33, 1);
        hamburger.Hamburger();

        System.out.print("\nEnter number of salads: ");
        int saladAmount = userInput.nextInt();
        LunchOrder salad = new LunchOrder(2.00, 1, 11, 5);
        salad.Salad();

        System.out.print("\nEnter number of fries: ");
        int frenchFryAmount = userInput.nextInt();
        LunchOrder frenchFries = new LunchOrder(1.30, 11, 36, 4);
        frenchFries.FrenchFries();

        System.out.print("\nEnter number of sodas: ");
        int sodaAmount = userInput.nextInt();
        LunchOrder soda = new LunchOrder(0.95, 0, 38, 0);
        soda.Soda();


        double total = (hamburgerAmount * hamburger.price) + (saladAmount * salad.price) + (frenchFryAmount * frenchFries.price) + (sodaAmount * soda.price);

        System.out.print("Your order comes to: $" + total);
    }


}
```

Within the TestLunchOrder class I prompt the user for an int value of the number of each item they want. Then I create an instance of the LunchOrder class with the appropriate price, fat, carb, and fiber parameters for each food. Alongside this, I call the appropriate food object with its respective method. Finally I initialize and print the total value by multiplying the number of each item by its price and totaling everything together.

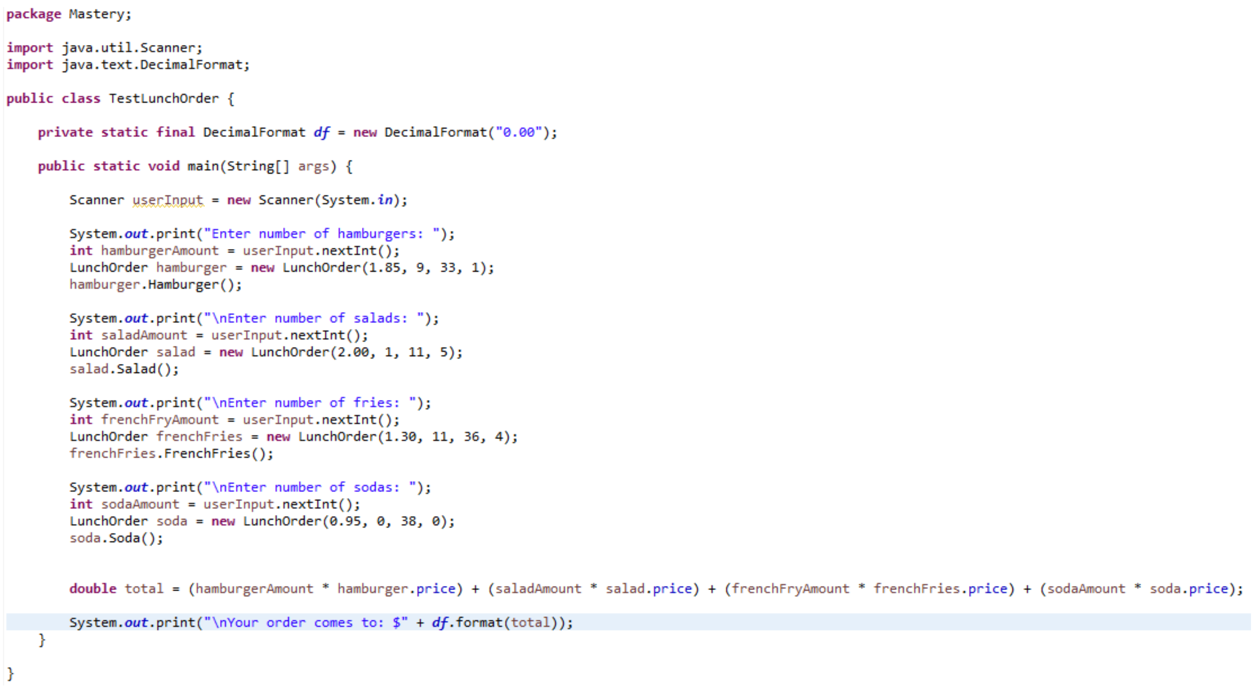
Below is the first rendition of the LunchOrder class:



```java
package skibidi;

public class LunchOrder {

     Double price;
     int fat;
     int carb;
     int fiber;

    LunchOrder(Double price, int fat, int carb, int fiber){

        this.price = price;
        this.fat = fat;
        this.carb = carb;
        this.fiber = fiber;

    }


    void Hamburger() {
        System.out.println("Each hamburger has " + this.fat + "g of fat, " + this.carb + "g of carbs, and " + this.fiber + "g of fiber.");
    }


    void Salad() {
        System.out.println("Each salad has " + this.fat + "g of fat, " + this.carb + "g of carbs, and " + this.fiber + "g of fiber.");
    }


    void FrenchFries() {
        System.out.println("French fries have " + this.fat + "g of fat, " + this.carb + "g of carbs, and " + this.fiber + "g of fiber.");
    }


    void Soda() {
        System.out.println("Each soda has " + this.fat + "g of fat, " + this.carb + "g of carbs, and " + this.fiber + "g of fiber.");
    }

}
```

First I declare the price, fat, carb, and fiber variables. Then within the constructor method I assign the variables with their respective values. Finally I create a method for each food that can print the fat, carb, and fiber amounts for their item. Looking back on the code I would most likely include another parameter being, that being ‘name’. That way I could eliminate the need to have a method for each food, and instead only have one that would use this.name in it.

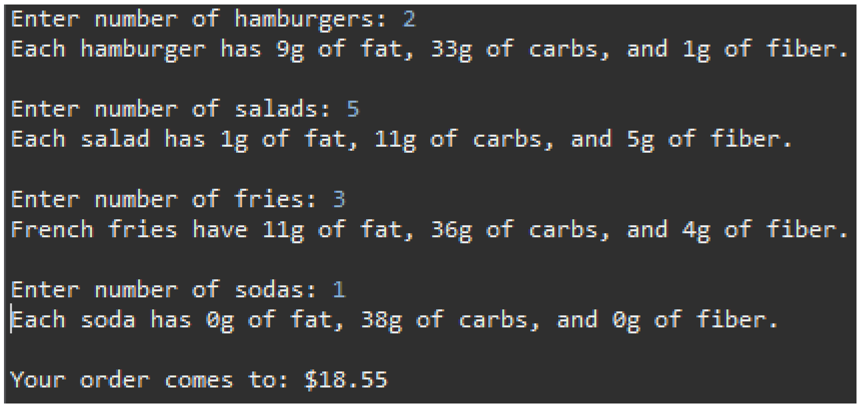
Below is the final rendition of the TestLunchOrder class:



```java
package Mastery;

import java.util.Scanner;
import java.text.DecimalFormat;

public class TestLunchOrder {

    private static final DecimalFormat df = new DecimalFormat("0.00");

    public static void main(String[] args) {

        Scanner userInput = new Scanner(System.in);

        System.out.print("Enter number of hamburgers: ");
        int hamburgerAmount = userInput.nextInt();
        LunchOrder hamburger = new LunchOrder(1.85, 9, 33, 1);
        hamburger.Hamburger();

        System.out.print("\nEnter number of salads: ");
        int saladAmount = userInput.nextInt();
        LunchOrder salad = new LunchOrder(2.00, 1, 11, 5);
        salad.Salad();

        System.out.print("\nEnter number of fries: ");
        int frenchFryAmount = userInput.nextInt();
        LunchOrder frenchFries = new LunchOrder(1.30, 11, 36, 4);
        frenchFries.FrenchFries();

        System.out.print("\nEnter number of sodas: ");
        int sodaAmount = userInput.nextInt();
        LunchOrder soda = new LunchOrder(0.95, 0, 38, 0);
        soda.Soda();


        double total = (hamburgerAmount * hamburger.price) + (saladAmount * salad.price) + (frenchFryAmount * frenchFries.price) + (sodaAmount * soda.price);

        System.out.print("\nYour order comes to: $" + df.format(total));
    }

}
```

The only new addition was the implementation of the DecimalFormat function, so that the total would be rounded to the hundreds place, preventing any sort of miscalculation.

Example of final output:



```
Enter number of hamburgers: 2
Each hamburger has 9g of fat, 33g of carbs, and 1g of fiber.

Enter number of salads: 5
Each salad has 1g of fat, 11g of carbs, and 5g of fiber.

Enter number of fries: 3
French fries have 11g of fat, 36g of carbs, and 4g of fiber.

Enter number of sodas: 1
Each soda has 0g of fat, 38g of carbs, and 0g of fiber.

Your order comes to: $18.55
```