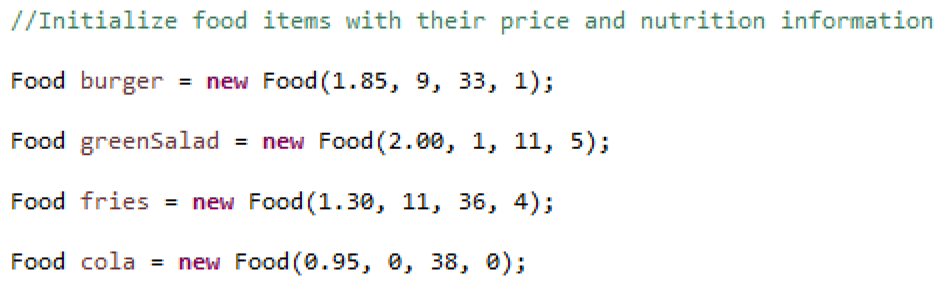
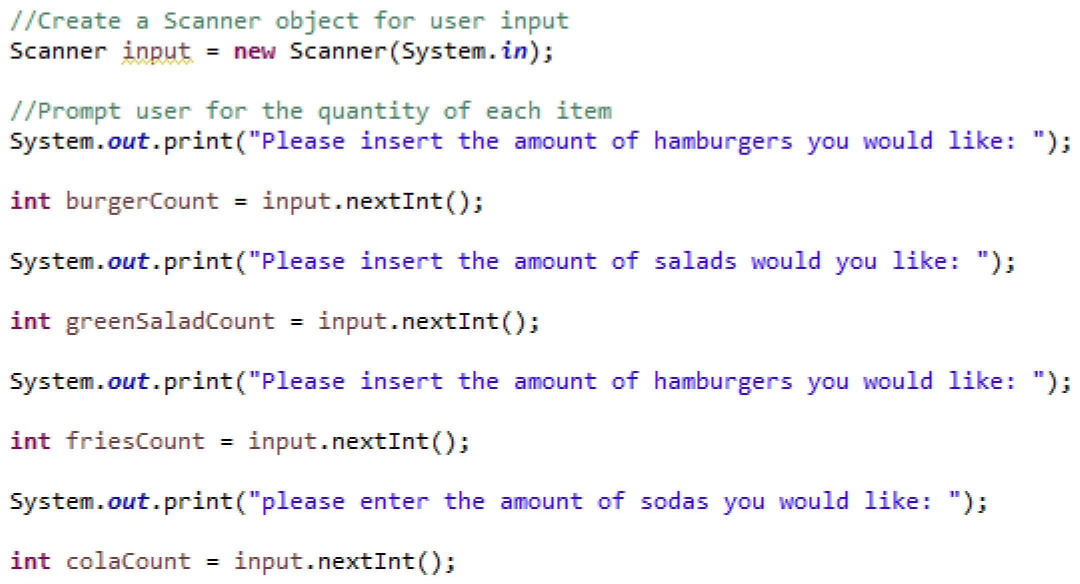
ReflectionLogs LunchOrder and Food

LunchOrder



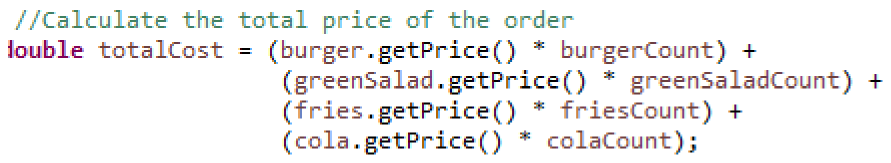
```
//Initialize food items with their price and nutrition information

Food burger = new Food(1.85, 9, 33, 1);

Food greenSalad = new Food(2.00, 1, 11, 5);

Food fries = new Food(1.30, 11, 36, 4);

Food cola = new Food(0.95, 0, 38, 0);
```

This code creates four instances of the Food class, each representing a different food item (burger, green salad, fries, and cola) with specific details. The constructor of the Food class is called with values for the price, fat, carbohydrates, and fiber content for each food item. These objects are then ready to be used later in the program for calculations or displaying nutritional information.



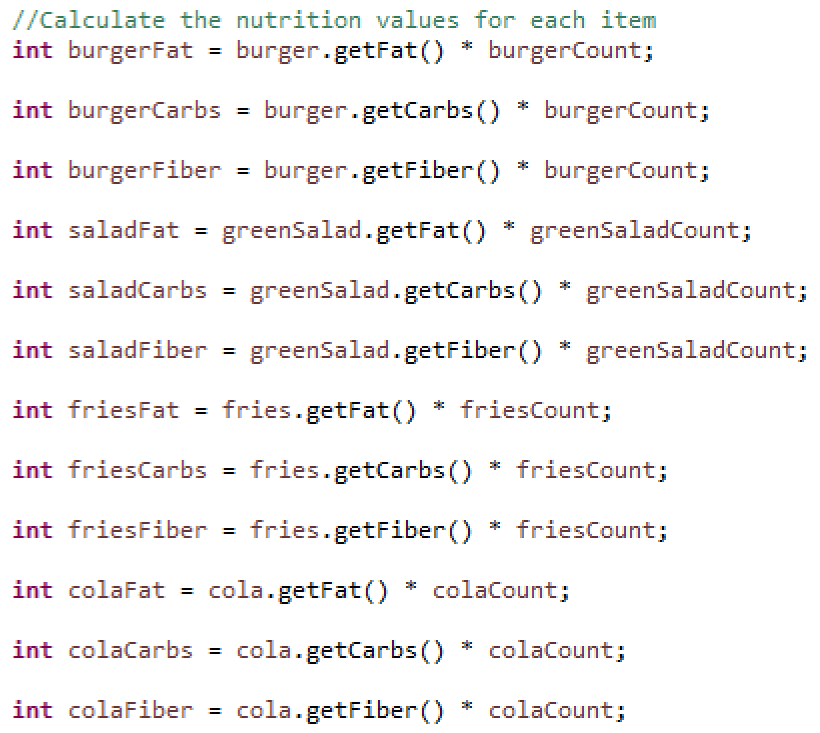
```java
//Create a Scanner object for user input
Scanner input = new Scanner(System.in);

//Prompt user for the quantity of each item
System.out.print("Please insert the amount of hamburgers you would like: ");

int burgerCount = input.nextInt();

System.out.print("Please insert the amount of salads would you like: ");

int greenSaladCount = input.nextInt();

System.out.print("Please insert the amount of hamburgers you would like: ");

int friesCount = input.nextInt();

System.out.print("please enter the amount of sodas you would like: ");

int colaCount = input.nextInt();
```

This code prompts the user to enter the quantity of different food items they would like to order. It uses a Scanner object to read user input for the number of hamburgers, salads, fries, and sodas. The user is asked to input the quantities, which are then stored in respective variables (burgerCount, greenSaladCount, friesCount, colaCount).



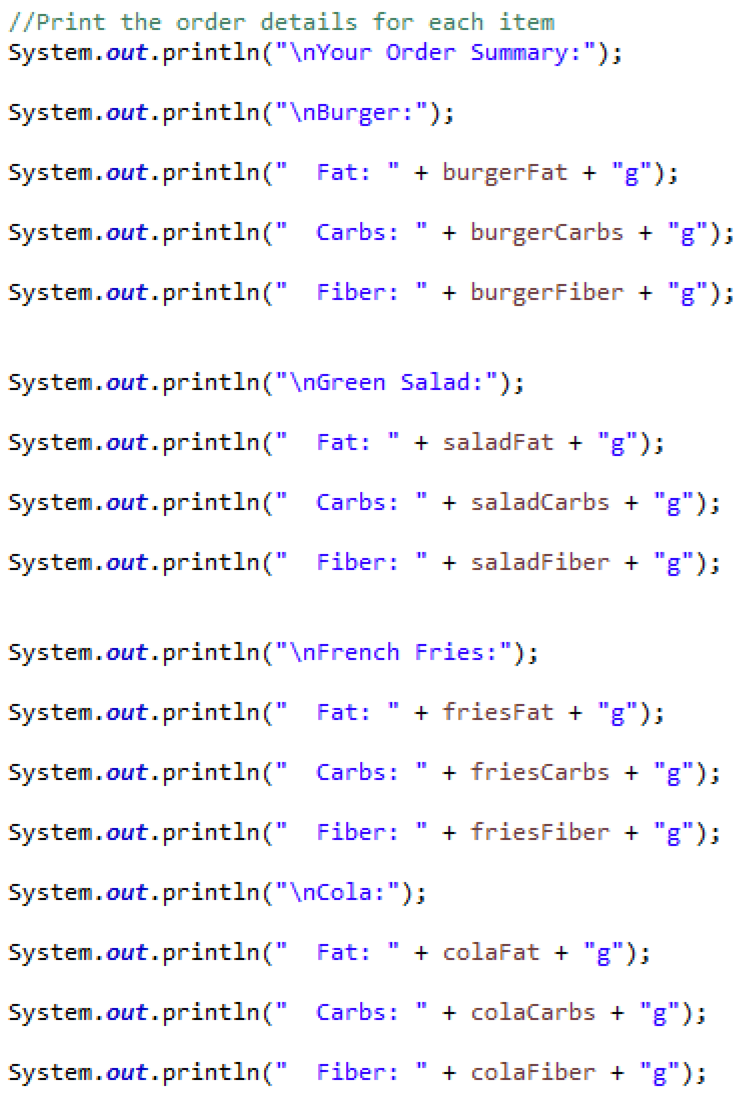
```java
 //Calculate the total price of the order
louble totalCost = (burger.getPrice() * burgerCount) +
                    (greenSalad.getPrice() * greenSaladCount) +
                    (fries.getPrice() * friesCount) +
                    (cola.getPrice() * colaCount);
```

This code calculates the total cost of the user's order by multiplying the price of each food item by the quantity selected. It does this for burgers, salads, fries, and sodas. The total cost is then computed by adding the individual costs for each item and stored in the totalCost variable.



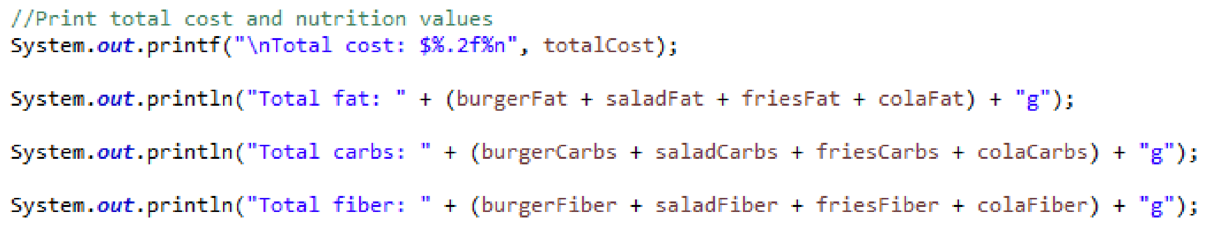
```
//Calculate the nutrition values for each item
int burgerFat = burger.getFat() * burgerCount;

int burgerCarbs = burger.getCarbs() * burgerCount;

int burgerFiber = burger.getFiber() * burgerCount;

int saladFat = greenSalad.getFat() * greenSaladCount;

int saladCarbs = greenSalad.getCarbs() * greenSaladCount;

int saladFiber = greenSalad.getFiber() * greenSaladCount;

int friesFat = fries.getFat() * friesCount;

int friesCarbs = fries.getCarbs() * friesCount;

int friesFiber = fries.getFiber() * friesCount;

int colaFat = cola.getFat() * colaCount;

int colaCarbs = cola.getCarbs() * colaCount;

int colaFiber = cola.getFiber() * colaCount;
```

This code calculates the total nutritional values (fat, carbohydrates, and fiber) for each item in the order by multiplying the respective nutritional values of each food item by the quantity the user has ordered. For example, burgerFat calculates the total fat content for the burgers by multiplying the fat per burger by the number of burgers. Similarly, it does this for carbohydrates and fiber for each food item (burger, salad, fries, and cola). The resulting values are stored in corresponding variables for fat, carbs, and fiber for each item.



```java
//Print the order details for each item
System.out.println("\nYour Order Summary:");

System.out.println("\nBurger:");

System.out.println("  Fat: " + burgerFat + "g");

System.out.println("  Carbs: " + burgerCarbs + "g");

System.out.println("  Fiber: " + burgerFiber + "g");


System.out.println("\nGreen Salad:");

System.out.println("  Fat: " + saladFat + "g");

System.out.println("  Carbs: " + saladCarbs + "g");

System.out.println("  Fiber: " + saladFiber + "g");


System.out.println("\nFrench Fries:");

System.out.println("  Fat: " + friesFat + "g");

System.out.println("  Carbs: " + friesCarbs + "g");

System.out.println("  Fiber: " + friesFiber + "g");

System.out.println("\nCola:");

System.out.println("  Fat: " + colaFat + "g");

System.out.println("  Carbs: " + colaCarbs + "g");

System.out.println("  Fiber: " + colaFiber + "g");
```

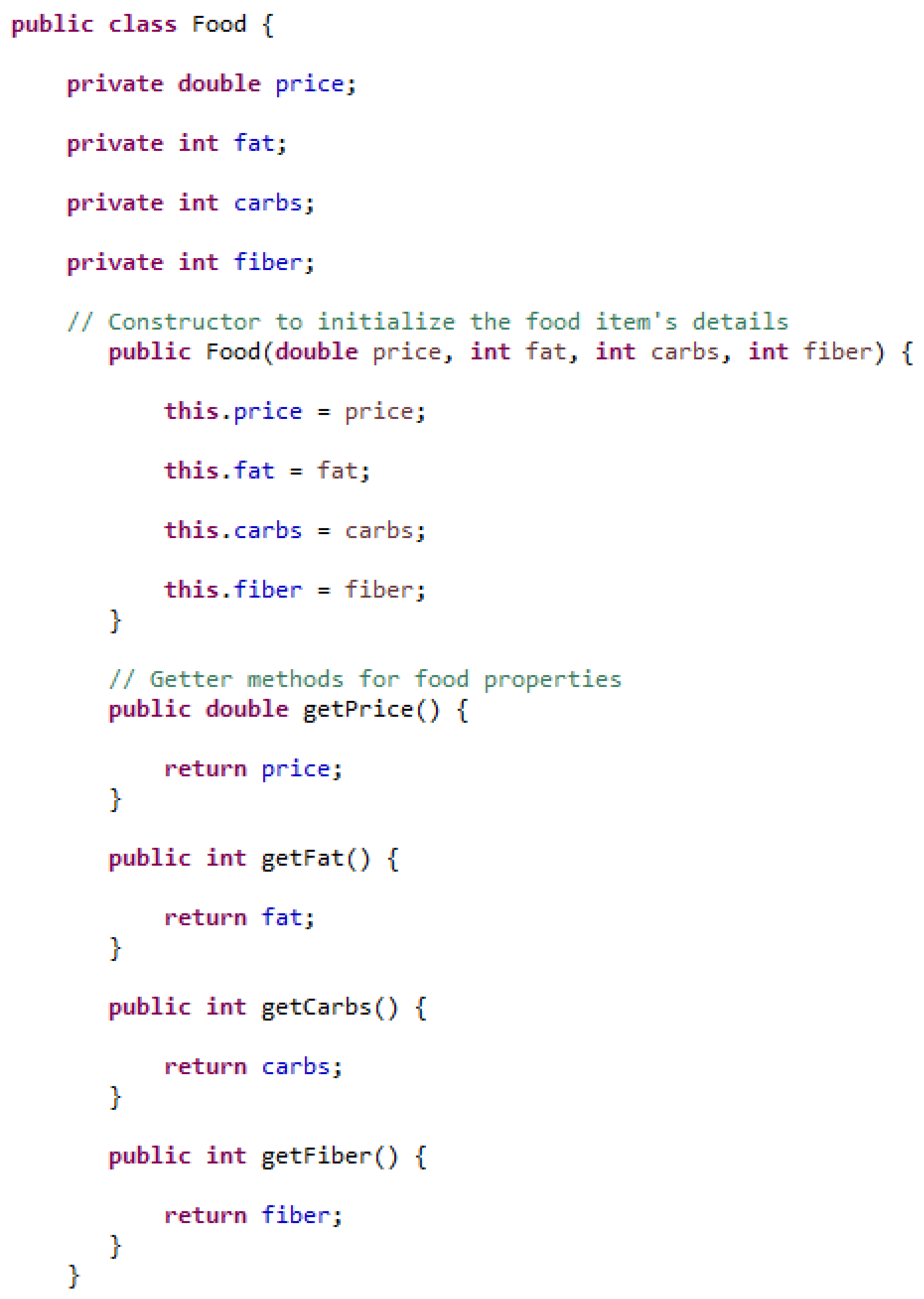
This code prints out the nutritional details (fat, carbohydrates, and fiber) for each food item in the order. It does this for burgers, green salads, French fries, and cola. For each item, it displays the total fat, carbs, and fiber based on the quantities ordered. Each nutritional value is printed on a new line under the corresponding food item, providing the user with a detailed order summary.



```
//Print total cost and nutrition values
System.out.printf("\nTotal cost: $%.2f%n", totalCost);

System.out.println("Total fat: " + (burgerFat + saladFat + friesFat + colaFat) + "g");

System.out.println("Total carbs: " + (burgerCarbs + saladCarbs + friesCarbs + colaCarbs) + "g");

System.out.println("Total fiber: " + (burgerFiber + saladFiber + friesFiber + colaFiber) + "g");
```

This code prints the total cost of the order and the overall nutritional values for fat, carbohydrates, and fiber. It displays the total cost with two decimal places using System.out.printf(). Then, it calculates and prints the total fat, carbs, and fiber by adding up the values from all food items (burger, salad, fries, and cola). Each of these totals is displayed in grams for fat, carbs, and fiber.

Food



```
public class Food {

    private double price;

    private int fat;

    private int carbs;

    private int fiber;

    // Constructor to initialize the food item's details
       public Food(double price, int fat, int carbs, int fiber) {

           this.price = price;

           this.fat = fat;

           this.carbs = carbs;

           this.fiber = fiber;
       }

       // Getter methods for food properties
       public double getPrice() {

           return price;
       }

       public int getFat() {

           return fat;
       }

       public int getCarbs() {

           return carbs;
       }

       public int getFiber() {

           return fiber;
       }
    }
```

This code defines a Food class with private instance variables for price, fat, carbs, and fiber, which represent the cost and nutritional information of a food item. The class includes a constructor that initializes these variables when a new Food object is created. It also provides getter methods (getPrice(), getFat(), getCarbs(), getFiber()) to allow access to the values of these private properties. These methods are used to retrieve the price, fat, carbohydrates, and fiber content of the food items. This structure

summarizes the food data, ensuring it can be accessed in a controlled manner. The class allows other parts of the program to interact with the food data without directly modifying its values.