CalPolyPomona

Chaitanya Ganpat Nalage

CS 5800 Homework - 04

Github Link: https://github.com/chaitanyanalage/CS5800

Question 1

Code:

```
package Assignment4.Q1;
import Assignment4.Q1.Pizza;

public class ConcreteBuilders {
    static class PizzaHutPizzaBuilder implements PizzaBuilder {
        private Pizza pizza;

        public PizzaHutPizzaBuilder() {
            this.pizza = new Pizza("Pizza Hut"); //creating a Pizza Hut pizza
        }

        @Override
        public void setSize(String size) {
            pizza.setSize(size);
        }

        @Override
        public void addPepperoni() {
            pizza.addTopping("Pepperoni");
        }

        @Override
        public void addSausage() {
            pizza.addTopping("Sausage");
        }

        @Override
        public void addMushrooms() {
            pizza.addTopping("Mushrooms");
        }

        @Override
        public void addBacon() {
            pizza.addTopping("Bacon");
        }

        @Override
        public void addOnions() {
            pizza.addTopping("Onions");
        }

        @Override
        public void addExtraCheese() {
            pizza.addTopping("Extra Cheese");
        }

        @Override
        public void addPeppers() {
            pizza.addTopping("Peppers");
```

```
        }

        @Override
        public void addChicken() {
            pizza.addTopping("Chicken");
        }

        @Override
        public void addOlives() {
            pizza.addTopping("Olives");
        }

        @Override
        public void addSpinach() {
            pizza.addTopping("Spinach");
        }

        @Override
        public void addTomatoAndBasil() {
            pizza.addTopping("Tomato and Basil");
        }

        @Override
        public void addBeef() {
            pizza.addTopping("Beef");
        }

        @Override
        public void addHam() {
            pizza.addTopping("Ham");
        }

        @Override
        public void addPesto() {
            pizza.addTopping("Pesto");
        }

        @Override
        public void addSpicyPork() {
            pizza.addTopping("Spicy Pork");
        }

        @Override
        public void addHamAndPineapple() {
            pizza.addTopping("Ham and Pineapple");
        }

        @Override
        public Pizza build() {
            return pizza;
        }
    }

    static class LittleCaesarsPizzaBuilder implements PizzaBuilder {
        private Pizza pizza;

        public LittleCaesarsPizzaBuilder() {
```

```
            this.pizza = new Pizza("Little Caesars"); //creating a Pizza Hut
pizza
        }

        @Override
        public void setSize(String size) {
            pizza.setSize(size);
        }

        @Override
        public void addPepperoni() {
            pizza.addTopping("Pepperoni");
        }

        @Override
        public void addSausage() {
            pizza.addTopping("Sausage");
        }

        @Override
        public void addMushrooms() {
            pizza.addTopping("Mushrooms");
        }

        @Override
        public void addBacon() {
            pizza.addTopping("Bacon");
        }

        @Override
        public void addOnions() {
            pizza.addTopping("Onions");
        }

        @Override
        public void addExtraCheese() {
            pizza.addTopping("Extra Cheese");
        }

        @Override
        public void addPeppers() {
            pizza.addTopping("Peppers");
        }

        @Override
        public void addChicken() {
            pizza.addTopping("Chicken");
        }

        @Override
        public void addOlives() {
            pizza.addTopping("Olives");
        }

        @Override
        public void addSpinach() {
            pizza.addTopping("Spinach");
```

```
        }

        @Override
        public void addTomatoAndBasil() {
            pizza.addTopping("Tomato and Basil");
        }

        @Override
        public void addBeef() {
            pizza.addTopping("Beef");
        }

        @Override
        public void addHam() {
            pizza.addTopping("Ham");
        }

        @Override
        public void addPesto() {
            pizza.addTopping("Pesto");
        }

        @Override
        public void addSpicyPork() {
            pizza.addTopping("Spicy Pork");
        }

        @Override
        public void addHamAndPineapple() {
            pizza.addTopping("Ham and Pineapple");
        }

        @Override
        public Pizza build() {
            return pizza;
        }
    }

    static class DominosPizzaBuilder implements PizzaBuilder {
        private Pizza pizza;

        public DominosPizzaBuilder() {
            this.pizza = new Pizza("Dominos"); //creating a Pizza Hut pizza
        }

        @Override
        public void setSize(String size) {
            pizza.setSize(size);
        }

        @Override
        public void addPepperoni() {
            pizza.addTopping("Pepperoni");
        }

        @Override
        public void addSausage() {
```

```
        pizza.addTopping("Sausage");
    }

    @Override
    public void addMushrooms() {
        pizza.addTopping("Mushrooms");
    }

    @Override
    public void addBacon() {
        pizza.addTopping("Bacon");
    }

    @Override
    public void addOnions() {
        pizza.addTopping("Onions");
    }

    @Override
    public void addExtraCheese() {
        pizza.addTopping("Extra Cheese");
    }

    @Override
    public void addPeppers() {
        pizza.addTopping("Peppers");
    }

    @Override
    public void addChicken() {
        pizza.addTopping("Chicken");
    }

    @Override
    public void addOlives() {
        pizza.addTopping("Olives");
    }

    @Override
    public void addSpinach() {
        pizza.addTopping("Spinach");
    }

    @Override
    public void addTomatoAndBasil() {
        pizza.addTopping("Tomato and Basil");
    }

    @Override
    public void addBeef() {
        pizza.addTopping("Beef");
    }

    @Override
    public void addHam() {
        pizza.addTopping("Ham");
    }
```

```
        @Override
        public void addPesto() {
            pizza.addTopping("Pesto");
        }

        @Override
        public void addSpicyPork() {
            pizza.addTopping("Spicy Pork");
        }

        @Override
        public void addHamAndPineapple() {
            pizza.addTopping("Ham and Pineapple");
        }

        @Override
        public Pizza build() {
            return pizza;
        }
    }

}
package Assignment4.Q1;
import Assignment4.Q1.Pizza;
import Assignment4.Q1.ConcreteBuilders.*;

public class DirectorDriver {
    public static void main(String[] args) {

        //Pizza Hut
        PizzaHutPizzaBuilder builder2_1_1= new PizzaHutPizzaBuilder();
        builder2_1_1.setSize("large");
        builder2_1_1.addPepperoni();
        builder2_1_1.addSausage();
        builder2_1_1.addOlives();
        Pizza pizza2_1_1 = builder2_1_1.build();
        pizza2_1_1.eat();

        PizzaHutPizzaBuilder builder2_1_2= new PizzaHutPizzaBuilder();
        builder2_1_2.setSize("small");
        builder2_1_2.addChicken();
        builder2_1_2.addPeppers();
        Pizza pizza2_1_2 = builder2_1_2.build();
        pizza2_1_2.eat();

        //Little Caesars
        LittleCaesarsPizzaBuilder builder2_2_1 = new
LittleCaesarsPizzaBuilder();
        builder2_2_1.setSize("medium");
        builder2_2_1.addExtraCheese();
        builder2_2_1.addMushrooms();
        builder2_2_1.addOlives();
        builder2_2_1.addOnions();
        builder2_2_1.addPeppers();
        builder2_2_1.addTomatoAndBasil();
        builder2_2_1.addBeef();
```

```
        builder2_2_1.addHam();
        Pizza pizza2_2_1 = builder2_2_1.build();
        pizza2_2_1.eat();

        LittleCaesarsPizzaBuilder builder2_2_2 = new
LittleCaesarsPizzaBuilder();
        builder2_2_2.setSize("small");
        builder2_2_2.addBacon();
        builder2_2_2.addChicken();
        builder2_2_2.addSpicyPork();
        builder2_2_2.addPeppers();
        builder2_2_2.addOnions();
        builder2_2_2.addHamAndPineapple();
        Pizza pizza2_2_2 = builder2_2_2.build();
        pizza2_2_2.eat();


        //Dominos
        DominosPizzaBuilder builder2_3_1 = new DominosPizzaBuilder();
        builder2_3_1.setSize("small");
        builder2_3_1.addHamAndPineapple();
        Pizza pizza2_3_1 = builder2_3_1.build();
        pizza2_3_1.eat();

        DominosPizzaBuilder builder2_3_2 = new DominosPizzaBuilder();
        builder2_3_2.setSize("large");
        builder2_3_2.addSausage();
        builder2_3_2.addMushrooms();
        builder2_3_2.addOlives();
        Pizza pizza2_3_2 = builder2_3_2.build();
        pizza2_3_2.eat();

    }
}

package Assignment4.Q1;

import java.util.ArrayList;
import java.util.List;

class Pizza {
    private String size;
    private String chain; // Add the pizza chain
    private List<String> toppings;

    // Constructor
    public Pizza(String chain) {
        this.chain = chain;
        this.toppings = new ArrayList<>();
    }

    //getters and setters
    public String getSize() {
        return size;
    }

    public void setSize(String size) {
```

```
        this.size = size;
    }

    public List<String> getToppings() {
        return toppings;
    }

    //method for adding toppings
    public void addTopping(String topping) {
        toppings.add(topping);
    }

    //eat method
    public void eat() {
        System.out.println(chain + " - Have fun and enjoy your " + size + "
pizza with your favorite toppings:");
        for (String topping : toppings) {
            System.out.println("        * " + topping);
        }
        System.out.println("-------------------------------------------------
-------------------------------------------------");
    }
} package Assignment4.Q1;

import Assignment4.Q1.Pizza;

public interface PizzaBuilder {
    void setSize(String size);
    void addPepperoni();
    void addSausage();
    void addMushrooms();
    void addBacon();
    void addOnions();
    void addExtraCheese();
    void addPeppers();
    void addChicken();
    void addOlives();
    void addSpinach();
    void addTomatoAndBasil();
    void addBeef();
    void addHam();
    void addPesto();
    void addSpicyPork();
    void addHamAndPineapple();

    Pizza build();
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java -javaagent:/Applications/I
Pizza Hut - Have fun and enjoy your large pizza with your favorite toppings:
        * Pepperoni
        * Sausage
        * Olives
-------------------------------------------------------------------------------------------
Pizza Hut - Have fun and enjoy your small pizza with your favorite toppings:
        * Chicken
        * Peppers
-------------------------------------------------------------------------------------------
Little Caesars - Have fun and enjoy your medium pizza with your favorite toppings:
        * Extra Cheese
        * Mushrooms
        * Olives
        * Onions
        * Peppers
        * Tomato and Basil
        * Beef
        * Ham
-------------------------------------------------------------------------------------------
Little Caesars - Have fun and enjoy your small pizza with your favorite toppings:
        * Bacon
        * Chicken
        * Spicy Pork
        * Peppers
        * Onions
        * Ham and Pineapple
-------------------------------------------------------------------------------------------
Dominos - Have fun and enjoy your small pizza with your favorite toppings:
        * Ham and Pineapple
-------------------------------------------------------------------------------------------
Dominos - Have fun and enjoy your large pizza with your favorite toppings:
        * Sausage
        * Mushrooms
        * Olives
-------------------------------------------------------------------------------------------

Process finished with exit code 0
```

Question 2

Code:

```
package Assignment4.Q2.factory;

import Assignment4.Q2.macronutrient.Carbs;
import Assignment4.Q2.macronutrient.Macronutrient;
import Assignment4.Q2.model.Customer;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class CarbsFactory implements MacronutrientFactory{
    private static CarbsFactory instance;
    private final List<String> allowedCarbs;

    private CarbsFactory() {
        allowedCarbs = new ArrayList<>(List.of("Cheese", "Bread", "Lentils",
"Pistachio"));
    }

    public static synchronized CarbsFactory getInstance() {
        if (instance == null) {
            instance = new CarbsFactory();
        }
        return instance;
    }

    @Override
    public Macronutrient create(Customer customer) {
        List<String> allowedCarbsCopy = new ArrayList<>(allowedCarbs);
        if (customer.getDietPlan().equals("No Restriction")) {
            allowedCarbsCopy = new ArrayList<>(allowedCarbs);
            Random random = new Random();
            String carb =
allowedCarbsCopy.get(random.nextInt(allowedCarbsCopy.size()));
            return new Carbs(carb);
        } else if (customer.getDietPlan().equals("Paleo")) {
            return new Carbs("Pistachio");
        } else if (customer.getDietPlan().equals("Vegan")) {
            //vegan meal regimen prohibits the consumption of dairy items,
such as cheese
            allowedCarbsCopy = new ArrayList<>(allowedCarbs);
            allowedCarbsCopy.remove("Cheese");
            Random random = new Random();
            String carb =
allowedCarbsCopy.get(random.nextInt(allowedCarbsCopy.size()));
            return new Carbs(carb);
        } else if (customer.getDietPlan().equals("Nut Allergy")) {
            //dietary regimen for nut allergies prohibits the consumption of
nuts, Pistachios included
            allowedCarbsCopy = new ArrayList<>(allowedCarbs);
            allowedCarbsCopy.remove("Pistachio");
            Random random = new Random();
```

```
            String carb =
allowedCarbsCopy.get(random.nextInt(allowedCarbsCopy.size()));
            return new Carbs(carb);
        } else {
            //manage alternative dietary regimens or ineffective dietary
strategies
            return null;
        }
    }

}

package Assignment4.Q2.factory;

import Assignment4.Q2.macronutrient.Fats;
import Assignment4.Q2.macronutrient.Macronutrient;
import Assignment4.Q2.model.Customer;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class FatsFactory implements MacronutrientFactory{
    private static FatsFactory instance;
    private final List<String> allowedFats;

    private FatsFactory() {
        allowedFats = new ArrayList<>(List.of("Avocado", "Sour cream",
"Tuna", "Peanuts"));
    }

    public static synchronized FatsFactory getInstance() {
        if (instance == null) {
            instance = new FatsFactory();
        }
        return instance;
    }

    @Override
    public Macronutrient create(Customer customer) {
        List<String> allowedFatsCopy = new ArrayList<>(allowedFats);
        if (customer.getDietPlan().equals("No Restriction")) {
            allowedFatsCopy = new ArrayList<>(allowedFats);
            Random random = new Random();
            String fat =
allowedFatsCopy.get(random.nextInt(allowedFatsCopy.size()));
            return new Fats(fat);
        } else if (customer.getDietPlan().equals("Paleo")) {
            //paleo diet regimen excludes all dairy items, such as Sour
cream, from its permitted foods
            allowedFatsCopy = new ArrayList<>(allowedFats);
            allowedFatsCopy.remove("Sour cream");
            Random random = new Random();
            String fat =
allowedFatsCopy.get(random.nextInt(allowedFatsCopy.size()));
            return new Fats(fat);
        } else if (customer.getDietPlan().equals("Vegan")) {
```

```
            //vegan meal plan prohibits the consumption of dairy items, such
as sour cream
            allowedFatsCopy = new ArrayList<>(allowedFats);
            allowedFatsCopy.remove("Sour cream");
            allowedFatsCopy.remove("Tuna");
            Random random = new Random();
            String fat =
allowedFatsCopy.get(random.nextInt(allowedFatsCopy.size()));
            return new Fats(fat);
        } else if (customer.getDietPlan().equals("Nut Allergy")) {
            //dietary regimen for those with a nut allergy prohibits the
consumption of nuts, including peanuts.
            allowedFatsCopy = new ArrayList<>(allowedFats);
            allowedFatsCopy.remove("Peanuts");
            Random random = new Random();
            String fat =
allowedFatsCopy.get(random.nextInt(allowedFatsCopy.size()));
            return new Fats(fat);
        } else {
            //manage alternative dietary regimens or ineffective dietary
strategies.
            return null;
        }
    }
}

package Assignment4.Q2.factory;

import Assignment4.Q2.model.Customer;

public class MacronutrientAbstractFactory {
    private static MacronutrientAbstractFactory instance;

    private MacronutrientAbstractFactory() {
        //constructor that is private to disallow instantiation from external
sources.
    }

    public static synchronized MacronutrientAbstractFactory getInstance() {
        if (instance == null) {
            instance = new MacronutrientAbstractFactory();
        }
        return instance;
    }

    public CarbsFactory createCarbsFactory() {
        return CarbsFactory.getInstance();
    }

    public ProteinFactory createProteinFactory() {
        return ProteinFactory.getInstance();
    }

    public FatsFactory createFatsFactory() {
        return FatsFactory.getInstance();
    }
}
```

```java
package Assignment4.Q2.factory;

import Assignment4.Q2.macronutrient.Macronutrient;
import Assignment4.Q2.model.Customer;

public interface MacronutrientFactory {
    Macronutrient create(Customer customer);
}
package Assignment4.Q2.factory;

import Assignment4.Q2.macronutrient.Macronutrient;
import Assignment4.Q2.macronutrient.Protein;
import Assignment4.Q2.model.Customer;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class ProteinFactory implements MacronutrientFactory{
    private static ProteinFactory instance;
    private final List<String> allowedProteins;

    private ProteinFactory() {
        allowedProteins = new ArrayList<>(List.of("Fish", "Chicken", "Beef",
"Tofu"));
    }

    public static synchronized ProteinFactory getInstance() {
        if (instance == null) {
            instance = new ProteinFactory();
        }
        return instance;
    }

    @Override
    public Macronutrient create(Customer customer) {
        List<String> allowedProteinsCopy = new ArrayList<>(allowedProteins);
        if (customer.getDietPlan().equals("No Restriction")) {
            allowedProteinsCopy = new ArrayList<>(allowedProteins);
            Random random = new Random();
            String protein =
allowedProteinsCopy.get(random.nextInt(allowedProteinsCopy.size()));
            return new Protein(protein);
        } else if (customer.getDietPlan().equals("Vegan")) {
            //vegan diet regimen prohibits the consumption of all meat items
            allowedProteinsCopy = new ArrayList<>(allowedProteins);
            allowedProteinsCopy.remove("Fish");
            allowedProteinsCopy.remove("Chicken");
            Random random = new Random();
            String protein =
allowedProteinsCopy.get(random.nextInt(allowedProteinsCopy.size()));
            return new Protein(protein);
        } else if (customer.getDietPlan().equalsIgnoreCase("Paleo")) {
            //paleo diet regimen limits specific protein options
            allowedProteinsCopy = new ArrayList<>(allowedProteins);
            allowedProteinsCopy.remove("Tofu");
            Random random = new Random();
```

```
            String protein =
allowedProteinsCopy.get(random.nextInt(allowedProteinsCopy.size()));
            return new Protein(protein);
        } else if (customer.getDietPlan().equalsIgnoreCase("Nut Allergy")) {
            //
            allowedProteinsCopy = new ArrayList<>(allowedProteins);
            Random random = new Random();
            String protein =
allowedProteinsCopy.get(random.nextInt(allowedProteinsCopy.size()));
            return new Protein(protein);
        }else {
            //dietary regimen for those with a nut allergy omits nuts
            return null;
        }
    }
}
package Assignment4.Q2.factory;

public class Singleton {
    private static Singleton instance;

    private Singleton() {
        //empty constructor
    }

    public static synchronized Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
package Assignment4.Q2.macronutrient;

public class Carbs implements Macronutrient {
    private String name;

    public Carbs(String name) {
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Carbs: " + name;
    }
}
package Assignment4.Q2.macronutrient;

public class Fats implements Macronutrient {
    private String name;

    public Fats(String name) {
```

```
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Fats: " + name;
    }
}
package Assignment4.Q2.macronutrient;

public interface Macronutrient {
    String getName();
}
package Assignment4.Q2.macronutrient;

public class Protein implements Macronutrient {
    private String name;

    public Protein(String name) {
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Protein: " + name;
    }
}
package Assignment4.Q2.meal;

import Assignment4.Q2.macronutrient.*;

public class MacronutrientMeal {
    private Carbs carbs;
    private Protein protein;
    private Fats fats;

    public MacronutrientMeal(Carbs carbs, Protein protein, Fats fats) {
        this.carbs = carbs;
        this.protein = protein;
        this.fats = fats;
    }

    public Carbs getCarbs() {
        return carbs;
    }

    public Protein getProtein() {
```

```
        return protein;
    }

    public Fats getFats() {
        return fats;
    }

    @Override
    public String toString() {
        return "Meal: " +
                "\n   * " + carbs +
                ",\n   * " + protein +
                ",\n   * " + fats +
                "\n----------------------------------------";
    }
}
package Assignment4.Q2.meal;

import Assignment4.Q2.factory.*;
import Assignment4.Q2.macronutrient.*;
import Assignment4.Q2.model.Customer;

public class MacronutrientMealFactory {
    private MacronutrientAbstractFactory abstractFactory;

    public MacronutrientMealFactory(MacronutrientAbstractFactory
abstractFactory) {
        this.abstractFactory = abstractFactory;
    }

    public MacronutrientMeal createMeal(Customer customer) {
        Carbs carbs = (Carbs)
abstractFactory.createCarbsFactory().create(customer);
        Protein protein = (Protein)
abstractFactory.createProteinFactory().create(customer);
        Fats fats = (Fats)
abstractFactory.createFatsFactory().create(customer);

        return new MacronutrientMeal(carbs, protein, fats);
    }
}
package Assignment4.Q2.model;

public class Customer {
    private String name;
    private String dietPlan;

    public Customer(String name, String dietPlan) {
        this.name = name;
        this.dietPlan = dietPlan;
    }

    public String getName() {
        return name;
    }

    public String getDietPlan() {
```

```
        return dietPlan;
    }
}
package Assignment4.Q2;

import Assignment4.Q2.factory.*;
import Assignment4.Q2.meal.*;
import Assignment4.Q2.model.Customer;

public class Driver {
    public static void main(String[] args) {
        MacronutrientAbstractFactory abstractFactory =
MacronutrientAbstractFactory.getInstance();
        MacronutrientMealFactory mealFactory = new
MacronutrientMealFactory(abstractFactory);

        // Create customers with different diet plans
        Customer customer1 = new Customer("Shreyas", "No Restriction");
        Customer customer2 = new Customer("Gokul", "Paleo");
        Customer customer3 = new Customer("Aashish", "Vegan");
        Customer customer4 = new Customer("Subham", "Nut Allergy");
        Customer customer5 = new Customer("Chaitanya", "Vegan");
        Customer customer6 = new Customer("Rashmi", "Paleo");

        // Generate meals for each customer
        MacronutrientMeal meal1 = mealFactory.createMeal(customer1);
        MacronutrientMeal meal2 = mealFactory.createMeal(customer2);
        MacronutrientMeal meal3 = mealFactory.createMeal(customer3);
        MacronutrientMeal meal4 = mealFactory.createMeal(customer4);
        MacronutrientMeal meal5 = mealFactory.createMeal(customer5);
        MacronutrientMeal meal6 = mealFactory.createMeal(customer6);

        // Print the generated meals
        System.out.println(customer1.getName() + ": Your " +
customer1.getDietPlan()+ " has " + meal1);
        System.out.println(customer2.getName() + ": Your " +
customer2.getDietPlan()+ " has " + meal2);
        System.out.println(customer3.getName() + ": Your " +
customer3.getDietPlan()+ " has " + meal3);
        System.out.println(customer4.getName() + ": Your " +
customer4.getDietPlan()+ " has " + meal4);
        System.out.println(customer5.getName() + ": Your " +
customer5.getDietPlan()+ " has " + meal5);
        System.out.println(customer6.getName() + ": Your " +
customer6.getDietPlan()+ " has " + meal6);
    }
}
```

1st Output:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Conte
Shreyas: Your No Restriction has Meal:
   * Carbs: Lentils,
   * Protein: Fish,
   * Fats: Avocado
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Fish,
   * Fats: Avocado
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Lentils,
   * Protein: Beef,
   * Fats: Peanuts
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Cheese,
   * Protein: Beef,
   * Fats: Sour cream
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Pistachio,
   * Protein: Tofu,
   * Fats: Peanuts
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Peanuts
----------------------------------------

Process finished with exit code 0
```

2nd Outputs:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Co
Shreyas: Your No Restriction has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Tuna
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Peanuts
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Lentils,
   * Protein: Tofu,
   * Fats: Avocado
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Bread,
   * Protein: Tofu,
   * Fats: Avocado
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------

Process finished with exit code 0
```

3rd Output:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/Co
Shreyas: Your No Restriction has Meal:
   * Carbs: Cheese,
   * Protein: Tofu,
   * Fats: Sour cream
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Chicken,
   * Fats: Tuna
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Bread,
   * Protein: Beef,
   * Fats: Tuna
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Bread,
   * Protein: Beef,
   * Fats: Peanuts
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Chicken,
   * Fats: Peanuts
----------------------------------------

Process finished with exit code 0
```

4th Output:

```
/Library/Java/JavaVirtualMachines/temurin-21.
Shreyas: Your No Restriction has Meal:
   * Carbs: Cheese,
   * Protein: Fish,
   * Fats: Peanuts
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Fish,
   * Fats: Avocado
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Bread,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Bread,
   * Protein: Tofu,
   * Fats: Sour cream
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Bread,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Chicken,
   * Fats: Peanuts
----------------------------------------

Process finished with exit code 0
```

5th Output:

```
/Library/Java/JavaVirtualMachines/temurin-21.jdk/
Shreyas: Your No Restriction has Meal:
   * Carbs: Pistachio,
   * Protein: Chicken,
   * Fats: Peanuts
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Bread,
   * Protein: Tofu,
   * Fats: Avocado
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Bread,
   * Protein: Tofu,
   * Fats: Avocado
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Bread,
   * Protein: Tofu,
   * Fats: Avocado
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Fish,
   * Fats: Peanuts
----------------------------------------

Process finished with exit code 0
```

6th Output:

```
/Library/Java/JavaVirtualMachines/temurin-21
Shreyas: Your No Restriction has Meal:
   * Carbs: Lentils,
   * Protein: Chicken,
   * Fats: Peanuts
----------------------------------------
Gokul: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Fish,
   * Fats: Peanuts
----------------------------------------
Aashish: Your Vegan has Meal:
   * Carbs: Pistachio,
   * Protein: Beef,
   * Fats: Avocado
----------------------------------------
Subham: Your Nut Allergy has Meal:
   * Carbs: Lentils,
   * Protein: Tofu,
   * Fats: Sour cream
----------------------------------------
Chaitanya: Your Vegan has Meal:
   * Carbs: Lentils,
   * Protein: Tofu,
   * Fats: Peanuts
----------------------------------------
Rashmi: Your Paleo has Meal:
   * Carbs: Pistachio,
   * Protein: Fish,
   * Fats: Tuna
----------------------------------------

Process finished with exit code 0
```