```
package com.mycompany.steppingstone5_recipetest;
/**
 * Recipe Test class
 *
 * Unit test for the Recipe class.
 * Creates a recipe object, gathers user input,
 * and prints the formatted recipe.
 */
public class SteppingStone5_RecipeTest {
    public static void main(String[] args) {
        SteppingStone5_Recipe recipe = new SteppingStone5_Recipe();
        recipe.createNewRecipe();
        recipe.printRecipe();
    }
}
```

```
package com.mycompany.steppingstone5_recipetest;

import java.util.ArrayList;
import java.util.Scanner;

/**
 * Recipe class
 *
 * Represents a recipe with a name, number of servings,
 * a list of ingredients, and total calories.
 *
 * Design decisions:
 * - ArrayList is used to store ingredients dynamically.
 * - Encapsulation is enforced through private variables
```

```
 *   with public accessors and mutators.
 */
public class SteppingStone5_Recipe {

    /* =========================
       Instance Variables
       ========================= */
    private String recipeName;
    private int servings;
    private ArrayList<Ingredient> recipeIngredients;
    private double totalRecipeCalories;

    /* =========================
       Constructors
       ========================= */

    // Default constructor
    public SteppingStone5_Recipe() {
        this.recipeName = "";
        this.servings = 0;
        this.recipeIngredients = new ArrayList<>();
        this.totalRecipeCalories = 0.0;
    }

    // Parameterized constructor
    public SteppingStone5_Recipe(String recipeName, int servings,
            ArrayList<Ingredient> recipeIngredients,
            double totalRecipeCalories) {
        this.recipeName = recipeName;
        this.servings = servings;
```

```
    this.recipeIngredients = recipeIngredients;
    this.totalRecipeCalories = totalRecipeCalories;
}

/* ========================
   Accessors and Mutators
   ======================== */

public String getRecipeName() {
    return recipeName;
}

public void setRecipeName(String recipeName) {
    this.recipeName = recipeName;
}

public int getServings() {
    return servings;
}

public void setServings(int servings) {
    this.servings = servings;
}

public ArrayList<Ingredient> getRecipeIngredients() {
    return recipeIngredients;
}

public void setRecipeIngredients(ArrayList<Ingredient> recipeIngredients) {
    this.recipeIngredients = recipeIngredients;
```

```
}

public double getTotalRecipeCalories() {
    return totalRecipeCalories;
}

public void setTotalRecipeCalories(double totalRecipeCalories) {
    this.totalRecipeCalories = totalRecipeCalories;
}

/* ==========================
   Methods
   ========================== */

/**
 * Prints the formatted recipe details.
 * Demonstrates object behavior and iteration.
 */
public void printRecipe() {
    System.out.println("\nRecipe: " + recipeName);
    System.out.println("Servings: " + servings);
    System.out.println("Ingredients:");

    for (Ingredient ingredient : recipeIngredients) {
        ingredient.printIngredient();
    }

    System.out.println("Total Calories: " + totalRecipeCalories);
    System.out.println("Calories per Serving: " +
            calculateCaloriesPerServing());
```

```
}

/**
 * Builds a recipe from user input.
 * Demonstrates Scanner usage, loops, and conditionals.
 */
public void createNewRecipe() {
    Scanner scnr = new Scanner(System.in);

    System.out.print("Enter recipe name: ");
    recipeName = scnr.nextLine();

    System.out.print("Enter number of servings: ");
    servings = scnr.nextInt();

    recipeIngredients = new ArrayList<>();
    totalRecipeCalories = 0.0;

    String addMore = "y";

    while (addMore.equalsIgnoreCase("y")) {

        System.out.print("Ingredient name: ");
        scnr.nextLine(); // clear buffer
        String name = scnr.nextLine();

        System.out.print("Amount: ");
        double amount = scnr.nextDouble();

        System.out.print("Unit of measure: ");
```

```
        scnr.nextLine(); // clear buffer
        String unit = scnr.nextLine();

        System.out.print("Calories: ");
        double calories = scnr.nextDouble();

        Ingredient ingredient =
                new Ingredient(name, amount, unit, calories);

        recipeIngredients.add(ingredient);
        totalRecipeCalories += calories;

        System.out.print("Add another ingredient? (y/n): ");
        addMore = scnr.next();
    }
}

/**
 * Custom Method
 *
 * Pseudocode:
 * IF servings > 0
 *     caloriesPerServing = totalRecipeCalories / servings
 * ELSE
 *     caloriesPerServing = 0
 * RETURN caloriesPerServing
 *
 * @return calories per serving
 */
public double calculateCaloriesPerServing() {
```

```
        if (servings > 0) {
            return totalRecipeCalories / servings;
        } else {
            return 0;
        }
    }
}
```

```
package com.mycompany.steppingstone5_recipetest;

/**
 * Ingredient class
 *
 * Represents a single ingredient in a recipe.
 * Stores the ingredient name, amount, unit of measure,
 * and calorie count.
 *
 * Design decisions:
 * - Encapsulation is enforced using private variables
 * - Public getters and setters allow controlled access
 */
public class Ingredient {

    /* ==========================
       Instance Variables
       ========================== */
    private String name;
    private double amount;
    private String unit;
    private double calories;
```

```
/* ==========================
   Constructors
   ========================== */

// Default constructor
public Ingredient() {
    this.name = "";
    this.amount = 0.0;
    this.unit = "";
    this.calories = 0.0;
}

// Parameterized constructor
public Ingredient(String name, double amount, String unit, double calories) {
    this.name = name;
    this.amount = amount;
    this.unit = unit;
    this.calories = calories;
}

/* ==========================
   Accessors and Mutators
   ========================== */

public String getName() {
    return name;
}

public void setName(String name) {
```

```
    this.name = name;
}

public double getAmount() {
    return amount;
}

public void setAmount(double amount) {
    this.amount = amount;
}

public String getUnit() {
    return unit;
}

public void setUnit(String unit) {
    this.unit = unit;
}

public double getCalories() {
    return calories;
}

public void setCalories(double calories) {
    this.calories = calories;
}

/* ==========================
   Methods
   ========================== */
```

```
    /**
     * Prints the formatted ingredient details.
     * This method replaces the previously unimplemented
     * method that caused a runtime exception.
     */
    public void printIngredient() {
        System.out.println("- " + amount + " " + unit + " of " + name
                + " (" + calories + " calories)");
    }
}
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.steppingstone5_recipetest;
import java.util.ArrayList;
import java.util.Scanner;
/**
 *
 * @author Darren J. Oates
 */
public class SteppingStone6_RecipeBox {

    /* =========================
     Instance Variables
     ========================= */
    private ArrayList<SteppingStone5_Recipe> listOfRecipes;
```

```
/* ============================
   Constructors
   ============================ */

// Default constructor
public SteppingStone6_RecipeBox() {
    this.listOfRecipes = new ArrayList<>();
}

// Constructor with parameters
public SteppingStone6_RecipeBox(ArrayList<SteppingStone5_Recipe> listOfRecipes) {
    this.listOfRecipes = listOfRecipes;
}

/* ============================
   Accessors and Mutators
   ============================ */

public ArrayList<SteppingStone5_Recipe> getListOfRecipes() {
    return listOfRecipes;
}

public void setListOfRecipes(ArrayList<SteppingStone5_Recipe> listOfRecipes) {
    this.listOfRecipes = listOfRecipes;
}

/* ============================
   Custom Methods
   ============================ */
```

```
/**
 * Prints the names of all recipes in the recipe box.
 */
public void printAllRecipeNames() {
    System.out.println("\nRecipe List:");
    for (SteppingStone5_Recipe recipe : listOfRecipes) {
        System.out.println("- " + recipe.getRecipeName());
    }
}

/**
 * Prints all details for a recipe that matches the given name.
 *
 * @param recipeName name of the recipe to display
 */
public void printAllRecipeDetails(String recipeName) {
    boolean found = false;

    for (SteppingStone5_Recipe recipe : listOfRecipes) {
        if (recipe.getRecipeName().equalsIgnoreCase(recipeName)) {
            recipe.printRecipe();
            found = true;
        }
    }

    if (!found) {
        System.out.println("Recipe not found.");
    }
}
```

```
/**
 * Creates a new recipe using user input and
 * adds it to the recipe box.
 */
public void addNewRecipe() {
    SteppingStone5_Recipe newRecipe = new SteppingStone5_Recipe();
    newRecipe.createNewRecipe();
    listOfRecipes.add(newRecipe);
}

/* =========================
   Main Method (Driver)
   ========================= */

public static void main(String[] args) {
    Scanner scnr = new Scanner(System.in);
    SteppingStone6_RecipeBox recipeBox = new SteppingStone6_RecipeBox();

    String choice;

    do {
        System.out.println("\nRecipe Manager Menu");
        System.out.println("1. Add new recipe");
        System.out.println("2. View all recipe names");
        System.out.println("3. View recipe details");
        System.out.println("4. Exit");
        System.out.print("Choose an option: ");

        choice = scnr.nextLine();
```

```
            switch (choice) {
                case "1" -> recipeBox.addNewRecipe();

                case "2" -> recipeBox.printAllRecipeNames();

                case "3" -> {
                    System.out.print("Enter recipe name: ");
                    String name = scnr.nextLine();
                    recipeBox.printAllRecipeDetails(name);
                }

                case "4" -> System.out.println("Exiting Recipe Manager.");

                default -> System.out.println("Invalid selection.");
            }

        } while (!choice.equals("4"));
    }
}
```