Southern New Hampshire University

IT 511 Milestone Two Guidelines and Rubric
The Recipe Class

Overview: In your final project, you will create a program that will help you manage a collection of recipes. The Recipe class you build for this milestone will hold all the details of the recipe, the methods to create a new recipe, and a method to print a recipe. In your final project submission, this class will also contain a custom method to add a new feature. In your submission for Milestone Two, you will include commented out pseudocode for this method.

Prompt: In this milestone, you submit the final project version of your Recipe class. Your submission should include the Recipe.java file and a Recipe_Test.java file.

Your Recipe class should include the following items:

- Instance variables: recipeName, servings, recipeIngredients, and totalRecipeCalories
- Accessors and mutators for the instance variables
- Constructors
- A printRecipe() method
- A createNewRecipe() method to build a recipe from user input
- Pseudocode for the custom method selected from the list in Stepping Stone Lab Five

Your Recipe_Test.java file containing a main() method that:

- Uses a constructor to create a new recipe
- Accesses the printRecipe() method to print the formatted recipe
- Invokes the createNewRecipe() method to accept user input

Specifically, the following **critical elements** of the final project are addressed:

I. **Data Types:** Your Recipe class should properly employ each of the following data types that meet the scenario's requirements where necessary:
   A. Utilize appropriate **numerical and string** data types to represent values for variables and attributes in your program.
   B. Populate a **list or array** that allows the management of a set of values as a single unit in your program.

II. **Algorithms and Control Structure:** Your Recipe class should properly employ each of the following control structures as required or defined by the scenario where necessary:
   A. Utilize **expressions or statements** that carry out appropriate actions or that make appropriate changes to your program's state as represented in your program's variables.
   B. Employ the appropriate **conditional control structures** that enable choosing between options in your program.
   C. Utilize **iterative control structures** that repeat actions as needed to achieve the program's goal.

III. **Methods:** Your Recipe class should properly employ each of the following aspects of method definition as determined by the scenario's requirements where necessary:
   A. Use **formal parameters** that provide local variables in a function's definition.
   B. Use **actual parameters** that send data as arguments in function calls.
   C. Create both **value-returning and void functions** to be parts of expressions or stand-alone statements in your program.
   D. Invoke methods that **access the services provided** by an object.
   E. Describe a **user-defined method** that provides custom services for an object.
   F. Create **unit tests** that ensure validity of the methods.

IV. **Classes:** Construct classes for your program that include the following as required by the scenario where necessary:
   A. Include **attributes** that allow for encapsulation and information hiding in your program.
   B. Include appropriate methods that provide an object's **behaviors**.

V. **Documentation:** Utilize **inline comments** directed toward software engineers for the ongoing maintenance of your program that explain the decisions you made in the construction of the classes in your program.

Rubric

Guidelines for Submission: Your complete program should be submitted as a zip file of the exported project containing the Recipe.java and Recipe_Test.java files.

Critical Elements	Proficient (100%)	Needs Improvement (80%)	Not Evident (0%)	Value
Data Types: Numerical and String	Utilizes appropriate numerical and string data types that represent values for variables and attributes in the program, meeting the scenario's requirements	Utilizes appropriate numerical and string data types that represent values for variables and attributes, but use of data types is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not utilize numerical and string data types that represent values for variables and attributes in the program	6
Data Types: List or Array	Populates a list or array that allows the management of a set of values as a single unit in the program, meeting the scenario's requirements	Populates a list or array that allows the management of a set of values as a single unit in the program, but population is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not populate a list or array that allows the management of a set of values as a single unit in the program	6

Algorithms and Control Structures: Expressions or Statements	Utilizes expressions or statements that carry out appropriate actions or that make appropriate changes to the program's state as represented in the program's variables and meet the scenario's requirements	Utilizes expressions or statements that carry out actions or that make changes to the program's state as represented in the program's variables, but use of expressions or statements is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not utilize expressions or statements that carry out actions or that make changes to the program's state as represented in the program's variables	6
Algorithms and Control Structures: Conditional Control Structures	Employs the appropriate conditional control structures, as the scenario defines, that enable choosing between options in the program	Employs the conditional control structures that enable choosing between options in the program, but use of conditional control structures is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's definition	Does not employ the conditional control structures that enable choosing between options in the program	7
Algorithms and Control Structures: Iterative Control Structures	Utilizes iterative control structures that repeat actions as needed to achieve the program's goal as required by the scenario	Utilizes iterative control structures that repeat actions to achieve the program's goal, but use of iterative control structures is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not utilize iterative control structures that repeat actions to achieve the program's goal	7
Methods: Formal Parameters	Uses formal parameters that provide local variables in a function's definition as determined by the scenario's requirements	Uses formal parameters that provide local variables in a function's definition, but use of formal parameters is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not use formal parameters that provide local variables in a function's definition	7
Methods: Actual Parameters	Uses actual parameters that send data as arguments in function calls as determined by the scenario's requirements	Uses actual parameters that send data as arguments in function calls, but use of actual parameters is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not use actual parameters that send data as arguments in function calls	7
Methods: Value-Returning and Void Functions	Creates both value-returning and void functions to be parts of expressions or stand-alone statements in the program as determined by the scenario's requirements	Creates both value-returning and void functions to be parts of expressions or stand-alone statements in the program, but functions are incomplete or illogical, contain inaccuracies, or lack accordance with the scenario's requirements	Does not create both value-returning and void functions to be parts of expressions or stand-alone statements in the program	7

Methods: Access Services Provided	Invokes methods that access the services provided by an object as required by the scenario	Invokes methods that access the services provided by an object, but called methods are incomplete or illogical, contain inaccuracies, or lack accordance with the scenario's requirements	Does not invoke methods that access the services provided by an object	7
Methods: User-Defined Methods	Employs user-defined methods that provide custom services for an object as specified in the program requirements	Employs user-defined methods that provide custom services for an object, but use of user-defined methods is incomplete or illogical, contains inaccuracies, or lacks accordance with the specifications in the program requirements	Does not employ user-defined methods that provide custom services for an object	9
Methods: Unit Tests	Creates unit tests that ensure validity of the methods as required by the scenario	Creates unit tests that ensure validity of the methods, but unit tests are incomplete or illogical, contain inaccuracies, or lack accordance with the scenario's requirements	Does not create unit tests that ensure validity of the methods	7
Classes: Attributes	Includes attributes, as required by the scenario, that allow for encapsulation and information hiding in the program	Includes attributes that allow for encapsulation and information hiding in the program, but inclusion is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not include attributes that allow for encapsulation and information hiding in the program	7
Classes: Behaviors	Includes appropriate methods that provide an object's behaviors, as required by the scenario	Includes methods that provide an object's behaviors, but inclusion of methods is incomplete or illogical, contains inaccuracies, or lacks accordance with the scenario's requirements	Does not include methods that provide an object's behaviors	7
Inline Comments	Utilizes inline comments directed toward software engineers for the ongoing maintenance of the program that explain the decisions made in the construction of the classes in the program	Utilizes inline comments that explain the decisions made in the construction of the classes in the program, but inline comments are incomplete or illogical, contain inaccuracies, or lack applicability towards software engineers for the ongoing maintenance of the program	Does not utilize inline comments that explain the decisions made in the construction of the classes in the program	10
			Total	**100%**