Name: _____ Lab Section: 02

CS161 Fall 2015 Programming Exam

Write your name above and return this paper to your lab instructor when you submit your exam. You must finish and submit your work using Check-In by the end of this lab.

Do your own work. Only look at your own computer screen. Do not browse, email, or use cell phones and other digital technology other than the computers in the lab.

During the exam you may use your Java text book (or ebook version if you let your instructor know), the Java API documentation website and the course website, but no other web pages.

Your grade will be based on:

- Correctness of each method
- Clarity of code and comments. Your code should be easy to read, and if there are non-obvious steps, they should be explained with comments.
- Object-oriented design: Does your code capture the idea of encapsulation
- Testing: Does your main method adequately test all aspects of your code?

You will write five java classes for keeping track of your recipes you like to cook. The names of the classes are Cook, Recipe, Breakfast, Lunch, and Dinner The classes Breakfast, Lunch, and Dinner must extend the class Recipe, to allow an instance of Cook to use a single ArrayList to hold instances of Breakfast, Lunch, and Dinner.

Your classes must implement the following methods.

Cook: class that is the primary application that you will run.

[**3 points**] **constructor:** Creates an empty ArrayList for holding items of type Recipe.

[**4 points**] **addRecipe:** Add a new Recipe, either a Breakfast, Lunch, or Dinner.

[**4 points**] **toString:** Returns a string representing all Recipes that have been added, by implicitly calling the toString method of each. Use `"\n"` in the String to make a new line for each Recipe.

[**3 points**] **main:** A main method that performs the test specified below.

Recipe: parent class of Breakfast, Lunch, and Dinner.

[**4 points**] **constructor:** Has three arguments, two Strings for a Recipe name and the source of the recipe, and a boolean value that is true if you have cooked this recipe. Assigns these values to protected instance variables. It also creates an empty ArrayList of Strings in a protected instance variable for keeping track of ingredients for this recipe.

[**3 points**] **getName:** Returns name.

[**3 points**] **setName:** Changes the value of name.

[**3 points**] **getSource:** Returns the source.

[**3 points**] **setSource:** Changes the value of the source.

[**3 points**] **getCooked:** Returns true if you have cooked this recipe.

[**3 points**] **setCooked:** Changes the value of the cooked instance variable.

[**4 points**] **getIngredients:** Returns the ArrayList of ingredients for this recipe.

[**4 points**] **addIngredient:** Adds an ingredient, given as a String, to this recipe's ingredients.

[**4 points**] **toString:** Returns a string like `Recipe: Spaghetti Cookbook Cooked` or `Recipe: Spaghetti Cookbook Not-Cooked`.

[**3 points**] **main:** A main method that tests each method in this class.

[**3 points**] **Breakfast:** child class of Recipe

[**3 points**] **constructor:** Has three arguments, two Strings for the recipe name and source, and a boolean value that is true if you have cooked this. Call the parent class's constructor to do all the work.

[**3 points**] **toString:** Returns a string like `Breakfast: Eggs Original Not-Cooked`

[**2 points**] **main:** A main method that tests each method in this class.

[**3 points**] **Lunch:** child class of Recipe

[**3 points**] **constructor:** Has three arguments, two Strings for the recipe name and source, and a boolean value that is true if you have cooked this. Call the parent class's constructor to do all the work.

[**3 points**] **toString:** Returns a string like `Lunch: PB-Sandwich Net Cooked`

[**2 points**] **main:** A main method that tests each method in this class.

[**3 points**] **Dinner:** child class of Recipe

[**3 points**] **constructor:** Has three arguments, two Strings for the recipe name and source, and a boolean value that is true if you have cooked this. Call the parent class's constructor to do all the work.

[**3 points**] **toString:** Returns a string like `Dinner: Beef-Stroganoff Baking-Beef Cooked`

[**2 points**] **main:** A main method that tests each method in this class.

Remember, you are required to implement a `main` method in every class that tests the methods of that class. After your tests in the main methods of each class work, change the main method in your Cook class to perform the following steps.

[**4 points**] Compiles.

1. Add these recipes in the order given.
   - A Lunch recipe named "Salad" taken from your "Martha Stewart Cookbook" that you have not cooked yet, with ingredients "lettuce" and "tomatoes".
   - A Dinner recipe named "Bacon-Potato" taken from your "Eatin Cheap Book" that you have cooked, with ingredients "potatoes", "bacon", and "onions".
   - A Breakfast recipe named "Pancakes" taken from your "Moosewood Cookbook" that you have not cooked, with ingredients "flour", "eggs", "butter", and "maple syrup".
   - A Dinner recipe named "Pizza" taken from "Dominoes" that you have not cooked, with ingredients "telephone" and "dollars".

[**5 points**] 2. Print the recipes using the toString method in Cook.

[**5 points**] 3. Make a BreakFast recipe using the information contained in your "Bacon-Potato" recipe, obtained using the appropriate get methods.

[**2 points**] 4. Print the recipes again using the toString method in Cook.

Here are some reminders about how to use ArrayLists.

- Remember to `import java.utils.ArrayList;`
- Constructing an instance: `ArrayList<Type> list = new ArrayList<Type>();`
- Add an object `o` of type Type doing `list.add(o)`
- Get element at index `i` by doing `list.get(i)`
- The `indexOf(x)` method returns the index of the first occurrence of object `x` in the list, determined by the `equals` method for the type of `x`. It returns -1 if it does not occur in the list.

When you are done, combine your files into a **jar file** named `cook.jar`

```
jar cvf cook.jar *.java
```

and check-in your jar file using the Checkin page at CS161 web site. And you must turn in this exam form with your name on it to your instructor before you leave recitation.