**LAB #1 - Creating Java Applications with Eclipse IDE**

**Due Date: Week 4 (demo of your code in class)**

**Value: 5%**

**Maximum points: 15 points**

**Purpose:** The purpose of this Lab assignment is to:

1. Become familiar with Eclipse IDE
2. Create simple Java Applications
3. Create and use a simple Java classes

**References:** Read the course’s text “Java How to program, 10th edition”, chapters 1 to 3 and the lecture notes. This material provides the necessary information that you need to complete the exercises.

**1 point**

**Overall Instructions**: Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students. You will have to demonstrate your solution in a scheduled lab session and submit the project **through dropbox link on eCentennial**.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228\_sectionNumber\_LabNumber**

Example: **JohSmith\_COMP228\_003\_Lab1**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228\_sectionNumber\_LabNumber.zip**

Example: **JohSmith\_COMP228\_003\_Lab1.zip**

Take screenshots of your app. Save them in the word document. The name of the file must be:

**YourFullName\_COMP228\_sectionNumber\_LabNumber**.doc

Example: **JohSmith\_COMP228\_003\_Lab1.doc**

**1 point**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

**Exercise #1: (8 points)**

Create a shell(blueprint) **– 3 points**

1. Create a public class named *Dish*.
2. Create some instance variables to hold the following information:
3. *String name,*
4. *String description,*
5. *integer servings,*
6. *you can add more variables if you want*
7. Provide all *getter* and *setter* methods
8. Provide a *getDishInformation* method to return information in a nicely formatted string. Use this method to display the information back to the user.
9. Provide a *default* *constructor*

Write a driver class **– 3 points**

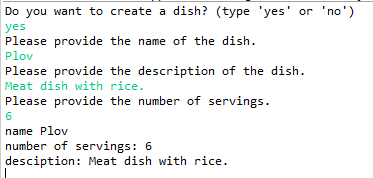
1. Create a driver class - Driver with main method
2. Instantiate a dish object.
3. Prompt user to enter all values for the dish and store them in the dish. See screenshots below.
4. If user declines to provide the dish data then display corresponding message. See screenshots below.

Use the *console* for input/output in the driver class. **– 1 point**

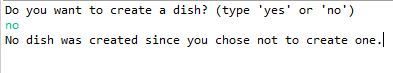
Internal documentation: **– 1 point**

1. Use *traditional comments* inside your class and driver class to provide explanation of what do method and other statements do (at least 4 comments)
2. Use *Javadoc comment* at the beginning of each class with your name as author and short description of the class

**Example:**



**Example:**



**Exercise #2: (5 points)**

Create a shell(blueprint) **– 2 points**

1. Create a class named *Recipe*.
2. Create some fields to hold the following information:
3. *ingredients*,
4. *preparation*,
5. Provide *getters* only.
6. Create a *multi-argument* *constructor* that sets the values.

Update Dish class: - **1 point**

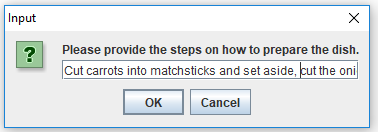
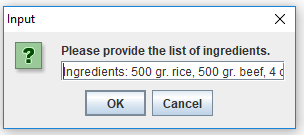
1. include Recipe as an instance variable
2. provide setter and getter for recipe in Dish class

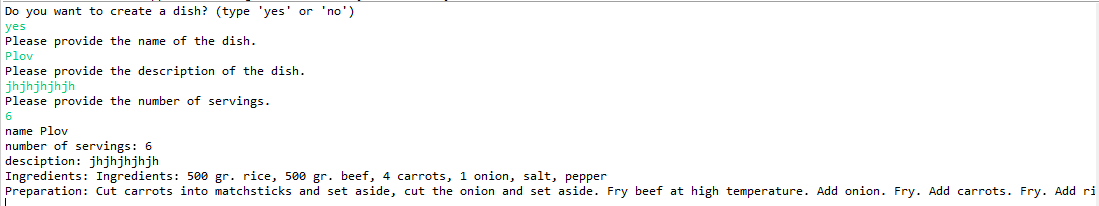
Update driver class: - **1 point**

1. add prompts to ask user to provide recipe information. See screenshots below.
2. Read the input from the user, populate information to the recipe in the dish using constructor, and then display the whole dish back to the user. See screenshots below.

Use the methods of *JOptionPane* class to accept input in the driver class. **– 1 points**

**Example**





**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** | **%** |
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods, etc.) | 35% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 45% |
| Comments, correct naming of variables, methods, classes, etc. | 10% |
| GUI | 10% |
| **Total** | **100%** |

Late submissions:

20% deducted for each day late.