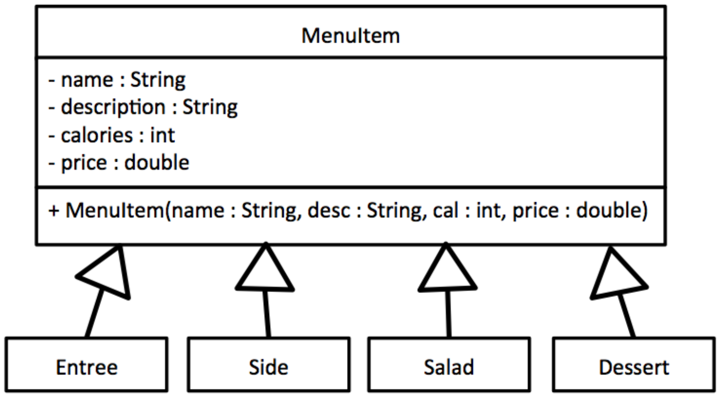
INFSCI 0017 – Fall 2018 **Final Project - MenuManager with Inheritance & GUI**

We are continuing the MenuManager project by adding inheritance and a Graphical User Interface (GUI) to our MenuManager project.

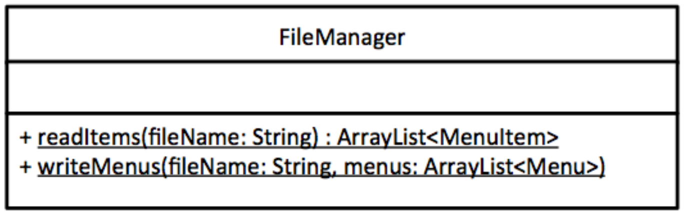
1. **[5 points] Do not create another project**, work in your project ***[your pitt id]_MenuManager*** (from Assignment 3). If you lost points for or had errors in the Assignment 3, you can see the solution posted in CourseWeb and fix it. Note that the solution for Assignment 3 has all the code but not all the comments. If you use the posted solution to start from, you should add the comments as indicated in Assignment 3.

2. **[18 points]** Class **MenuItem** is defined as the superclass for Entree, Side, Salad and Dessert (all these 4 classes extend MenuItem). As a result, the properties and getters/setters previously defined are moved to MenuItem.
   - See the following diagram. Implement the class MenuItem.



   - Make Entree, Side, Salad and Dessert classes extend MenuItem (as indicated in the figure) and remove from them all properties and Getters/Setters. The properties name, description, calories are now inherited.
   - Add the property ***price*** and the corresponding getter and setter in the class MenuItem. Now all other classes extending MenuItem also inherit *price*.
   - In all classes Entree, Side, Salad and Dessert make constructors to receive all property values and to call the super constructor.
   - Override toString() in MenuItem in order to return the name attribute.
   - Also, override toString() in class Menu (this class is defined in previous Assignment 4) in order to return the name of the menu.

3. **[12 points]** Modify **FileManager** class. See the figure below.


FileManager
+ readItems(fileName: String) : ArrayList<MenuItem>
+ writeMenus(fileName: String, menus: ArrayList<Menu>)

   - The following method

     ```
     public static ArrayList<MenuItem> readItems(String fileName)
     ```

     reads all types of dishes from a single file in which each line can be an entree, a side, a salad or a dessert. TIP: take one of the methods implemented before, such as readEntrees, and modify it.

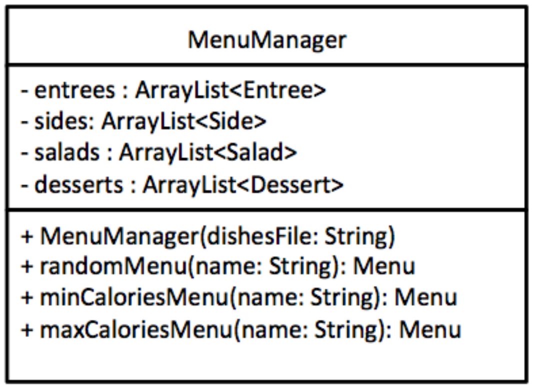
     The format of a line in file dishes.txt is

     ```
     name of dish@@type of the dish@@description of the dish@@calories@@price
     ```

     Where "type of the dish" is either "entree", or "side", or "salad", or "dessert" (Take a look to the file included *data/dishes.txt*). This "type of the dish" determines what kind of object to create, which class to use: Entree, Side, Salad, or Dessert. Then the object is added to an ArrayList<MenuItem>. Since all these classes extend MenuItem, objects of them can be "treated" generically as MenuItem objects.

- The method
  ```
  public static void writeMenu( String fileName, ArrayList<Menu> menus )
  ```
  writes a file (use String fileName parameter) with the information of **all** the menus in the ArrayList<Menu> menus. You are free of choosing the format in which the data is written, but for each Menu in the ArrayList, all information should be included: name of the menu, name of each dish, description of each dish, calories and price of each dish, plus the sum of all calories, and the total price.

4. **[15 points]** Implement the class **MenuManager**. A MenuManager object has an ArrayList of each of the type of dishes (like MenuRandomize in Assignment 3).



MenuManager

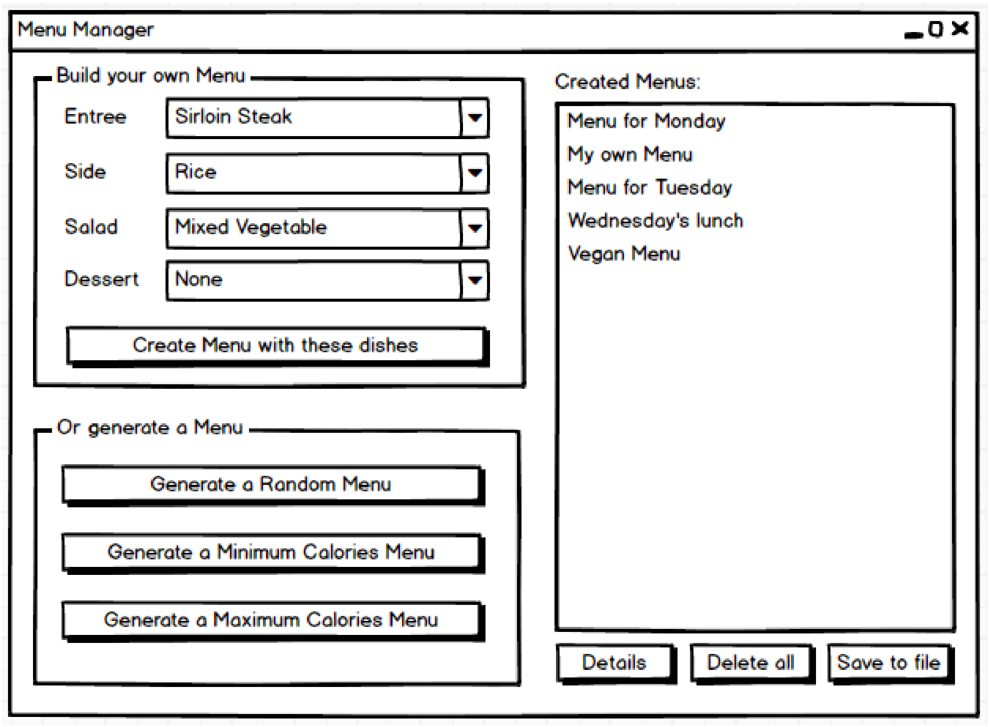
- entrees : ArrayList<Entree>
- sides: ArrayList<Side>
- salads : ArrayList<Salad>
- desserts : ArrayList<Dessert>

+ MenuManager(dishesFile: String)
+ randomMenu(name: String): Menu
+ minCaloriesMenu(name: String): Menu
+ maxCaloriesMenu(name: String): Menu

- The **constructor**, as you can see in the following figure, receives only one fileName. You can not use the old methods in FileManager because now, all dishes are in a single file. Read this file using the method *readItems* of the class FileManager and fill a single ArrayList of MenuItem. Now we need to separate the single ArrayList containing MenuItem objects into the four ArrayList of different types. To solve this, create the four ArrayList and implement a loop go through the single ArrayList<MenuItem> and take every dish and put it in the right ArrayList. Hint: user ***instanceof*** operator. You *can* implement this task in a separate method and call it in the constructor.
- Integrate the method *randomMenu* from old class MenuRandomize into MenuManager. The method creates a Menu object taking randomly one entree, one side, one salad, and one dessert. The name of the the menu is passed in the parameter *String name*, which a difference from the previous method implemented in Assignment 3.
- **[Optional: extra credit]** Methods minCaloriesMenu() and maxCaloriesMenu() generates the lowest and highest calories menus, respectively. To do the minimum, you have to pick the Entree with the lowest calorie value among entrees. Same for side, salad and dessert. The method maxCaloriesMenu() do similarly, but selecting highest calories dishes.
- Add getters and setters to all properties in MenuManager.

5. **[45 points] Graphical User Interface**. Build the following GUI.

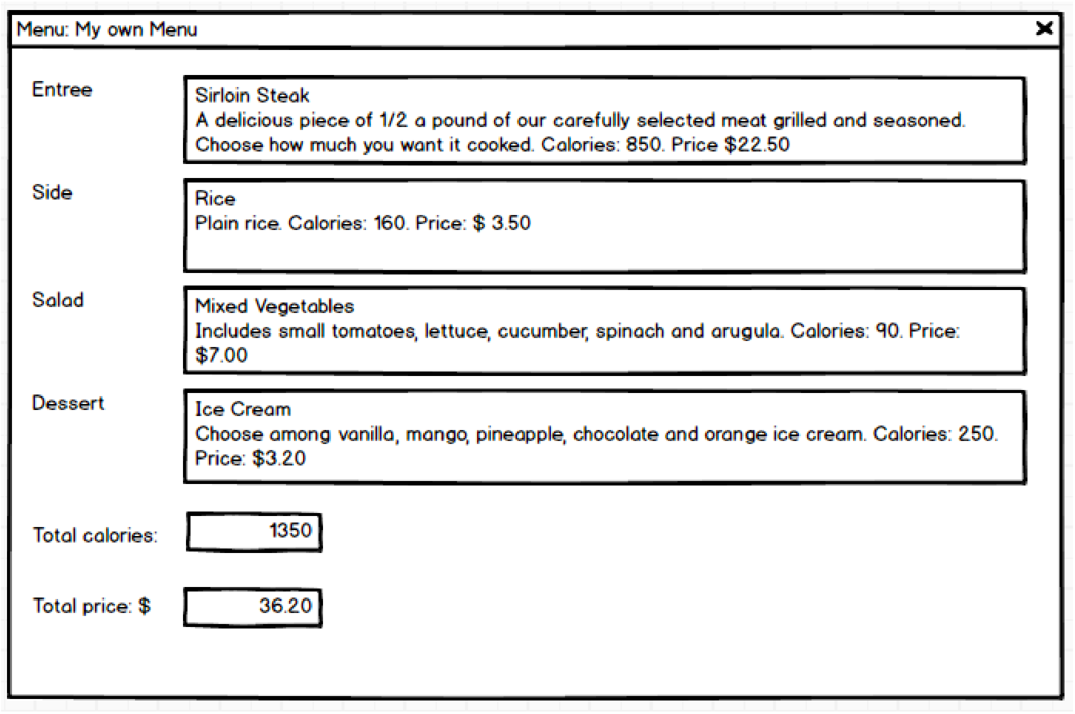
FIGURE 1. Main window implemented in class MenuManagerGUI



Entree: Sirloin Steak
Side: Rice
Salad: Mixed Vegetable
Dessert: None

Menu for Monday
My own Menu
Menu for Tuesday
Wednesday's lunch
Vegan Menu

FIGURE 2. Secondary window for displaying the details of a Menu, also implemented in MenuManagerGUI class



Entree	Sirloin Steak A delicious piece of 1/2 a pound of our carefully selected meat grilled and seasoned. Choose how much you want it cooked. Calories: 850. Price $22.50
Side	Rice Plain rice. Calories: 160. Price: $ 3.50
Salad	Mixed Vegetables Includes small tomatoes, lettuce, cucumber, spinach and arugula. Calories: 90. Price: $7.00
Dessert	Ice Cream Choose among vanilla, mango, pineapple, chocolate and orange ice cream. Calories: 250. Price: $3.20
Total calories:	1350
Total price: $	36.20

- Create the class ***MenuManagerGUI*** containing all graphic components and a MenuManager object. Also contains a main(String[] args) method.
- Declare ALL components (all JLabel, JFrame, JButton, JComboBox, JTextField, etc) as **properties** in *MenuManagerGUI*.
- Make sure you declare a MenuManager object as a property of the MenuManagerGUI class.
- The constructor of the class *MenuManagerGUI* should do:
    - Create the MenuManager object, which loads the data from the file.
    - Initialize and place all graphic components
    - Load the ArrayLists of MenuManager object into the comboboxes. In other words, fill the comboboxes in the main window with the ArrayList of entrees, sides, salads and desserts that are contained in the class MenuManager.
- The execution of the MenuManagerGUI.main method does the following:
    - Creates a MenuManagerGUI object calling the constructor explained before
    - Set the JFrame visible, so the main window appear
- The main window in the GUI gives you four options to generate Menu objects. These four options correspond to the buttons:
    - "Create Menu with these dishes" button take the selected elements in the four comboboxes and creates a menu. Just before create the menu object, it ask the user to input a name for the menu. The newly created menu is added to the list at the side.
    - "Generate a Random Menu": this button ask the user to input a name for the new menu and then uses the method randomMenu in class MenuManager. The random menu generated is then added to the list in the right side.
    - **[Optional: extra credit]** "Generate a Minimum Calories Menu": similarly than the previous button but now using the method *minCaloriesMenu*() from MenuManager.
    - **[Optional: extra credit]** "Generate a Maximum Calories Menu": similarly than the previous button but now using the method *maxCaloriesMenu*() from MenuManager.
- The list of the right side is where generated menus are placed. The list shows the menus by their names. For achieving this, make sure you override the method toString in class Menu.
- The button Details (FIGURE 2) displays the secondary window filling all the fields contained there (all textfields and textareas) with the information of the selected Menu in the list of generated menus in the main window (right side in FIGURE 1). Note that the name of the menu selected is in the title of the Details window (FIGURE 2). If no menu is selected in the list, then the secondary window is not shown.
- The button "Delete all" removes all Menu elements from the list.
- The button "Save to File" writes a file "data/menus.txt" with the whole data of the menus in the list. Use the method *writeMenus* from the FileManager class.
- All text fields in the secondary window (FIGURE 2) are "read only". This mean their values are filled by the program but the user can not change them.

6. **[Optional: extra credit] Documentation: add JAVA-DOC style COMMENTS and GENERATE the documentation inside a folder *doc*. Locate this folder inside your project folder, at the same level of folders "src" and "data". To do this you have to complete two steps:**
    - Write comments in a special format, as defined in Assignment 3. The official Java page where this format is explained is: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html. Another link with easy examples is http://www.tutorialspoint.com/java/java_documentation.htm
    - Run javadoc tool which generate the HTML documentation from the comments you wrote. In ECLIPSE this is easy: just go to the option "Generate Javadoc" in menu "Project".

7. **[5 points] Check and correct your INDENTATION.**

By completing the items marked as "[Optional: extra credit]" you will receive 3 extra credit for your course grade.
Export your project and compress it in a file named *[your_pittid]_FinalProject_INFSCI0017.zip*.
Due date is Thursday, December 6th 23:59 PM. Submit using CourseWeb submission tool.