ITIS/ITCS 4180/5180 Mobile Application Development
In Class Assignment 1

Basic Instructions:

---

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. **Export your project as follows:**
   a. From eclipse, choose *"Export…"* from the File menu.
   b. From the Export window, choose *General* then *File System*. Click *Next*.
   c. Make sure that your project for this assignment is selected. Make sure that all of its subfolders are also selected.
   d. Choose the location you want to save the exported project directory to. For example, your *Desktop* or *Documents* folder.
   e. When exporting make sure you select *Create directory structure for files*.
   f. Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the .java and resource files.
5. Submission details:
   a. When you submit the assignment, compress your exported project into a single zip file. The format of compressed file name is IClass_HW#.zip
   b. You should submit the assignment through Moodle: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

In Class Assignment 1 (100 Points)

In this assignment you will get familiar with Java's List Interface. You will also practice some Object Oriented techniques by creating your own Java class to use within your code. ***Your implementation should target the most efficient algorithms and data structures. You will be graded based on the efficiency of your implementation. You will not be awarded any points if you use simple nested loops to implement the below tasks.*** You should use the List interface, and you are encouraged to review the lecture slides and the Java documentation.

Part 1 (50 Points):

In the support files you are given the file "passwordDictionary.txt"[1], which includes a list of real world passwords. Each line of the file represents a single password. You are asked to perform the following tasks:

1. Create a class called PartOne.java, which should include the implementation for this part.
2. Read the passwords from the "passwordDictionary.txt" file. Each password record should to be assigned to a Password object.
3. The Password class should contain these instance variables, password (password String), length (number of characters) and score (integer).
4. The Password class should also contain a method called calculateScore(), this method calculates and sets the password score. The password score initialized to 0 and is incremented based on based on the following criteria:
   a. If the password length < 5 then increment the password score by 1.
   b. Else If the password length >= 5 and < 8 then increment the password score by 2.
   c. Else If the password length >= 8 then increment the password score by 3.
   d. If the password contains at least one uppercase alphabet letter then increment score by 1.
   e. If the password contains at least one lowercase alphabet letter then increment score by 1.
   f. If the password contains at least one special character in the following !, #, *, /, (, _, and % then increment score by 2.
   g. If the password contains at least one number from 0-9 then increment score by 2.
5. The below table shows a few examples of password scores using the above criteria:

Password	length	lowercase letter	uppercase letter	numeric digit	special character	score
!!!12Jps	(8 chars) +3	+1	+1	+2	+2	9
!2#4%6&8(0	(10 chars) +3	0	0	+2	+2	7
#adBanner!	(10 chars) +3	+1	+1	0	+2	7
663600	(6 chars) +2	0	0	+2	0	4

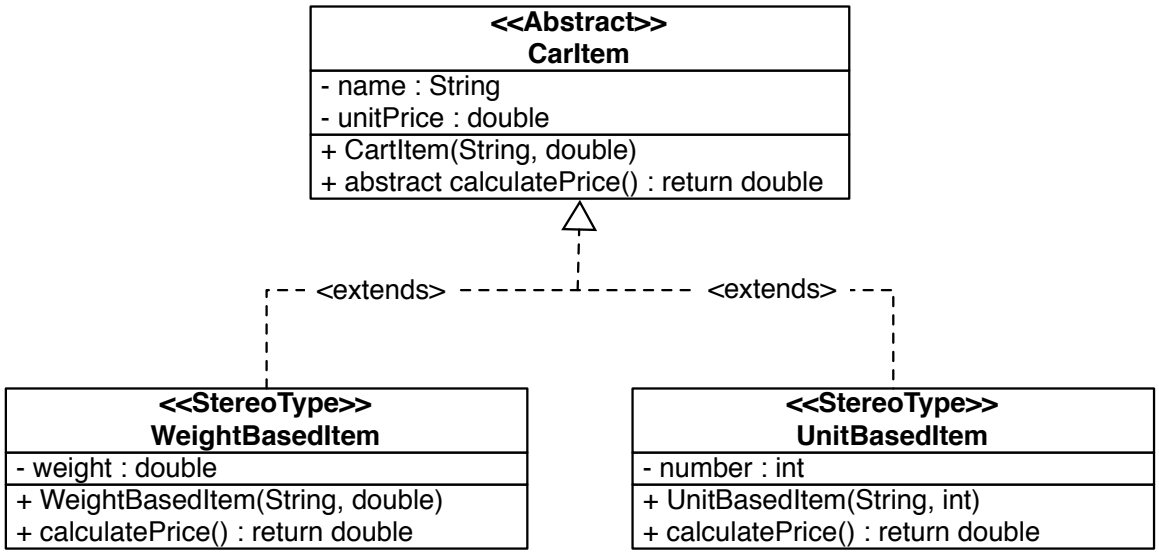
[1] The password file was retrieved from http://dazzlepod.com/site_media/txt/passwords.txt

6. We recommend using the Pattern class in java for pattern matching in password strings. http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
7. Store the password objects in a LinkedList and sort the passwords by their computed scores. Hint: To sort use the Collections.sort() method.
8. Print the top 10 passwords sorted by score. Then print the bottom 10 passwords sorted by score.

Part 2 (50 Points):

In this part, you will implement a shopping cart system. You are asked to implement this system using the Object-Oriented principles you've been introduced to through the video lectures. You are asked to perform the following tasks:

1. The **CartItem** class, it is an abstract class, that includes the item name and item unit price, and an abstract method to calculate the item's overall price. The item price is to be calculated according to the item type.
2. You should design two classes that extend the **CartItem** class namely the **WeightBasedItem** and **UnitBasedItem** classes. For the **WeightBasedItem** class the price is calculated based on the item's weight (lbs), while for the **UnitBasedItem** class the price is calculated based on the number of items. Design your classes based on the below class diagram. You can add extra methods if needed. To print an item you should override the toString() method for both the **WeightBasedItem** and **UnitBasedItem** classes.



3. To use these classes, you need to create **Cart** class which responsible for storing the cart items, adding items to the cart, printing the cart contents, and calculating the total price of the Cart based on the items added to the cart. The Cart should store and manage the items in a List. The Cart class should be implemented based on the below class diagram.

Cart
- items : ArrayList<CartItem>
+ Cart() + add(CartItem) + print()

4. You are provided with a file “cart_data.txt”, which includes an example items to be inserted in a cart. Each line of the file represents a single item. Each record consists of the item name, item type (UnitBased or WeightBased), number or weight of the item, and the price per unit or per pound. The values are comma separated, for example, Banana,UnitBased,20,0.59. Which is a banana, unit based item, and 20 bananas at 0.59/banana.
5. You are provided with a class called PartTwo, which should create the cart, parse the “cart_data.txt” file, populate the cart with the items retrieved from the file, calculate the total price of the cart based on the items added to the cart, and then print the cart contents and the cart overall price. Below is an example print out of a cart.

```
Name: Banana, Number: 20, Unit Price: $0.59, Total: $11.8
Name: Grape, Weight: 2.5, Unit Price: $1.33, Total: $15.125
.......
.......
------------------------------
Total: $100.5
```

The following code that reads in a file line by line. It is assumed the file is included in root folder of the Eclipse project. Use this code to help you read the provided files.

```
public void readFileAtPath(String filename) {
	// Lets make sure the file path is not empty or null
	if (filename == null || filename.isEmpty()) {
		System.out.println("Invalid File Path");
		return;
	}
	String filePath = System.getProperty("user.dir") + "/" + filename;
	BufferedReader inputStream = null;
	// We need a try catch block so we can handle any potential IO errors
	try {
		try {
			inputStream = new BufferedReader(new FileReader(filePath));
			String lineContent = null;
			// Loop will iterate over each line within the file.
			// It will stop when no new lines are found.
			while ((lineContent = inputStream.readLine()) != null) {
				System.out.println("Found the line: " + lineContent);
			}
		}
		// Make sure we close the buffered reader.
		finally {
			if (inputStream != null)
				inputStream.close();
		}
	} catch (IOException e) {
		e.printStackTrace();
	}
}// end of method
```

String Tokenization:
To split the contents of a single line read a file.

```
String[] resultingTokens = lineContent.split(",");
for (int i = 0; i < resultingTokens.length; i++){
		System.out.println(resultingTokens [i].trim());
}
```