COP 3330
Spring 2019
Final Term Assignment 1

Parking Garage System

Introduction: For this assignment, you will use Java and Eclipse. In this assignment we will create a system for parking garages at UCF. We will use the concept of abstract class, interface, string and file operations. The top of your ALL source files must contain the following course header (with correct assignment number) identifying you as an author:

```
/* University of Central Florida
 * COP 3330 Spring 2019
 * Assignment #1
 * Author: <Your Name>
 */
```

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Programming assignment rules mentioned in syllabus, are also applied in this submission.

Eclipse Implementation:

1. Create a Java project in Eclipse called "ParkingGarageSystem".

2. You will need to create 4 java classes in the “src” subfolder of the package.

    a. The ABSTRACT class would be Garage (Eclipse will put this class in the file “Garage.java”). This is a base class for garages.
    b. Next there would be one interface, ParkingSpot (Eclipse will put this class in the file “ParkingSpot.java”).
    c. The third class would be GarageC (Eclipse will put this class in the file “GarageC.java”). This class will inherit properties from Garage class and implement ParkingSpot interface.

3. Add the course header with your name at the top of the file.
4. Write your Java codes.
5. Save your Java codes.
6. Run your Java codes.
7. Submit only the four mentioned java source files on webcourses (Please DO NOT submit the compressed project or JAR file).
    a. Garage.java

b. ParkingSpot.java
c. GarageC.java
d. GarageTest.java

Provided Resources:

A text file “cars.txt”. You can use this text file to test your code, however we will be grading your submission with a DIFFERENT text file. There are tabs(“\t”) between the words and numbers. The test case text file will be of same format of the given cars.txt file, only with different car names, width, and length.

The Problem:

In this problem we will try to learn and implement abstract class and interface. We will also learn to do the String and File operations. We will create 3 classes to maintain the parking system of UCF. For the sake of simplicity, we will consider only three types of cars (irrespective of make and model): Small, Medium, Large. Your job is to read the text file containing all the car information at a time, and determine:

(i) How occupied the garage is,
(ii) How many different types of cars are present in the garage,
(iii) If the garage is still OPEN or FULL.

From implementational point of view, here are certain points which could be helpful:

1. The base class (which is also abstract) Garage has the following features:
    a. A private variable to take garage name
    b. A constructor for setting the garage name
    c. A getter method for garage name
    d. An abstract method to calculate the total occupied area for a garage. **Hint- the input argument is an Arraylist containing the areas of all the cars.**

2. The ParkingSpot interface has the following features. Note that the methods will not be implemented within the interface:
    a. A method specification which takes the width and length of a car and returns the type of the parking (Small/Medium/Large), needed for the car. The length range for those three types of cars are: Small (<=15 feet), Medium (> 15 but <=17), and Large (> 17).
    b. A method which will return the area of a parking spot, taking width and length of the car as arguments. Take the width and length as double.

3. The class GarageC will inherit all the properties from Garage and implement the all properties from ParkingSpot. It has some extra features:
    a. A private variable for number of floors. Type: Integer.
    b. A private variable for each floor area. Type: Double.

c. A **PRIVATE** constructor that will initialize the instance variables of GarageC. It will take three arguments, garage name, number of floors, and each floor area.
d. You have to implement Single tone design pattern. The whole system cannot have more than one GarageC object. So, create a factory method that utilize the singleton design pattern and prevent to have more than one object of GarageC.
e. A getter method for number of floors variable.
f. A getter method for each floor area variable.
g. Properly implement the interface methods and abstract method.

4. Create a GarageTest class and it should contain the main method. In the main method, please take three input for GarageC: one for number of floors of Garage C (point 3.a), second for each floor area of Garage C (point 3.b), and lastly the filename, (for example "cars.txt" file). You should keep the file in appropriate directory so that your program can read it.
    a. Please read the file using proper file operations.
    b. Perform the proper string operations and get width and length for each car.
    c. Determine parking spot's type for each car.
    d. Determine parking spot area for each car.
    e. While adding the cars in the garage from the file, determine the total area occupied in Garage C after reading each car. Also, and print the information in the console like this (Car: truck1, parking type: Large, Area: 150, Total area occupied in Garage C: 150.66, Total vacant area in Garage C: 500)
    f. Whenever, the Garage C becomes full, print it in the console "Garage C is full" and write the list of the car type and their count into a text file called "out.txt" (example: small: 3, medium: 2). To taste this part of the code, you need to provide a relatively smaller number of floor and area for each floor, so that the garage becomes full with the input data from the file.

Sample Input/Output Format:

Enter number of floor: 2
Enter area at each floor: 500
Enter input file name: cars.txt
Car: truck1, parking type: Large, Area: 150.66, Total area occupied in Garage C: 150, Total vacant area in Garage C: 850 //because 500*2 – 150
….
…..
If at any point the total area occupied exceeds 1000, it should display full and write out.txt with the information as require din in 4.f above.

What to Submit: Please submit only the Java source files in webcourses:

a. Garage.java
b. ParkingSpot.java
c. GarageC.java
d. GarageTest.java

Please make sure that everything compiles and run.

Grading: Your submission will be graded using the grading rubric below. Please remember, late submissions will get 0 point directly.

Criteria	Points
1. Create Garage.java abstract class and set each feature correctly.	1
2. Create ParkingSpot.java interface correctly.	1
3. Create GarageC class properly with proper inheritance and implementation	3
4. Reading the file properly and displaying the occupancy status of the garage.	3
5. Show whether the garage is OPEN/FULL	0.5
6. Writing the appropriate data into the out.txt file (if unable to write in the file, display in the console will result in partial point)	1.5

Happy Coding!!!