CS 250 - Fall 2021 - Homework 05

Objectives

The objectives of this assignment are:

- Testing your understanding of sorting algorithms
- To give additional practice with array and looping constructs.
- To give additional practice with String methods.
- To give additional practice with parsing inputs and exception handling.

Part 1. Non-programming Questions (50 points)

Q1. (30 points) Given the following array of integers: {45, 11, 50, 59, 60, 2, 4, 7, 10}.

Draw the figures to demonstrate how to apply selection sort, bubble sort, insertion sort, and merge sort to sort the above list. Use the example figure in the slides of lecture 10.

Q2. (20 pts) Insertion sort in descending order

Use the code of insertion sort in Lab 10 to modify and create a new version of insertion sort that sorts the array in descending order. Fill your code below:

```
private static void insertionSortDesc(int[] arrInt) {
    //TODO add code below
}
```

Part 2. Programming Question (50 points + bonus 10 points)

2.1 Problem Descriptions

Develop a Java application to read a text file containing cars' information into your program, including car ID, car make, car model, year, and license plate. Then, the users can sort the list of cars in different ways.

2.2 Problem Design and Implementation Guideline

2.2.1. Create the HW#5 Project and a text file names cars.txt:

First, you need to create a Java project HW5 on Eclipse. Then, follow the similar approach in Lab4 to create a text file named **cars.txt** under the **src** folder of your project. You can do that by right-clicking on the **src** item on the project explorer window. Then, choose **New**, **File**, enter **cars.txt** in the **File name** text box, and click the **Finish** button. A subwindow to edit the content of this file is opened.

This file contains the car records in separate rows. Each row stores one car's information, including **id**, **make**, **model**, **year**, and **license plate**. A comma separates the data fields of each student. Copy the following contents to this **cars.txt** file and save it.

```
201, Toyota, Camry, 1985, VRX-6224
302, Chevrolet, Chevette, 1980, SM-1243
103, Honda, Civic, 1993, RSS-8839
544, Ford, Mustang, 1966, AZU-4852
345, Dodge, Neon, 1996, LLC-0674
```

2.2.2. Car class: You need to create a Car class to store each car's information read from the above file. Code this class with the following requirements:

- This class has file instance variables: **ID** (int), **make** (String), **model** (String), **year** (int - year), and **license plate** (String)
- This class has the following methods:

+ **Car(int id, String make, String model, int year, String licencePlate)**: The constructor of this class initializes the instance variables with the parameters (assign parameters to corresponding instance variables)
+ Getter and Setter methods for each instance variables
+ **String toString()**: This method overrides the toString() method derived from the Object class. You should return a string including all car information, i.e., "id, make, model, year, license plate".

2.2.3. CarSorting class:

You should create a new class named CarSorting. Use the following skeleton code to start your coding of this class:

```
public class CarSorting {
    private static List<Car> listCars = new ArrayList<>();          //list of cars read from
file cars.txt

   private static void readCarsFile() {
      //TODO 1: write code to read file src/cars.txt and load the records into listCars
   }

    public static void main(String[] args) {
```

```
        // Read the cars.txt file to load the cars into listCars
        readCarsFile();
        // Sort the list of cars in different ways
        Scanner sc = new Scanner(System.in);
        while(true) {
            System.out.print("Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: ");
            int att = Integer.parseInt(sc.nextLine());
            //TODO 2: write code below
        }
    }
```

Implement the following tasks:

A. TODO1

First, you need to finish the method readCarsFile() that reads each line of the file cars.txt, parse intro multiple tokens, create the Car object and add to the array list listCars. Check lab 04 source code to implement this method.

B. TODO2

For the second task, in the main method, there is existing code of calling readCarsFile to load the cars information from file to the listCars variable. Then, a while loop has been constructed for you to keep asking the sort option in the att variable.

What you need to do is to write code to perform the tasks as follows:

- If att is 0, sort the listCars in the ascending order of the ID of the car.
- If att is 1, sort the listCars in the ascending order of the make of the car.
- If att is 2, sort the listCars in the ascending order of the model of the car.
- If att is 3, sort the listCars in the ascending order of the year of the car.
- If att is 4, sort the listCars in the ascending order of the license plate of the car.
- Else, print "Exit!" and quit the program
- After sorting listCars, print the list of cars to the Console (each car on a line)

Note: To implement sorting, you can either use the sort() method supported by Java or implement your own method based on four sorting methods we learned.

Below is the sample output of the program:

```
Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: 0
Car [103, Honda, Civic, 1993, RSS-8839]
Car [201, Toyota, Camry, 1985, VRX-6224]
Car [302, Chevrolet, Chevette, 1980, SM-1243]
Car [345, Dodge, Neon, 1996, LLC-0674]
Car [544, Ford, Mustang, 1966, AZU-4852]
Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: 1
Car [302, Chevrolet, Chevette, 1980, SM-1243]
```

```
Car [345, Dodge, Neon, 1996, LLC-0674]
Car [544, Ford, Mustang, 1966, AZU-4852]
Car [103, Honda, Civic, 1993, RSS-8839]
Car [201, Toyota, Camry, 1985, VRX-6224]
Enter sort option 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: 2
Car [201, Toyota, Camry, 1985, VRX-6224]
Car [302, Chevrolet, Chevette, 1980, SM-1243]
Car [103, Honda, Civic, 1993, RSS-8839]
Car [544, Ford, Mustang, 1966, AZU-4852]
Car [345, Dodge, Neon, 1996, LLC-0674]
Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: 3
Car [544, Ford, Mustang, 1966, AZU-4852]
Car [302, Chevrolet, Chevette, 1980, SM-1243]
Car [201, Toyota, Camry, 1985, VRX-6224]
Car [103, Honda, Civic, 1993, RSS-8839]
Car [345, Dodge, Neon, 1996, LLC-0674]
Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: 4
Car [544, Ford, Mustang, 1966, AZU-4852]
Car [345, Dodge, Neon, 1996, LLC-0674]
Car [103, Honda, Civic, 1993, RSS-8839]
Car [302, Chevrolet, Chevette, 1980, SM-1243]
Car [201, Toyota, Camry, 1985, VRX-6224]
Enter sort option: 0-ID, 1-Make, 2-Model, 3-Year, 4-LP: -1
Exist!
```

2.3 Submission Requirements

A. Source code: Write your **own** code

B. Program requirements: Your program can solve and meet all of the requirements described in 2.1 and 2.2. Do not write additional code that is not required.

C. Coding style: Your source code must be properly indented and contain adequate comments for class, methods, variables, etc.

- At the beginning of your Java class file, specify the following items:
  - Your name, the assignment number for this program, the creation date
  - The problem description that defines your application
  - Enumerate (list) the goals of your application
  - Enumerate (list) the inputs of your application
  - Enumerate (list) the outputs of your application
- Before each method in your class, specify the following items:
  - Specify the purpose of the corresponding method
  - Enumerate the possible inputs and outputs of the corresponding method

    - Write down the pseudocode to implement the corresponding method
- Others comments for variables, inside method implementation

2.4 Grading Rubrics

Your code in this part will be evaluated using the following grading scheme (50 points for completed functional source code + bonus 10 points for good code comments):

Points	Item
25	Correct class, variable, and method definition
10	Consistent coding-style, such as indents, curly bracket positions, etc.
15	Have all required variables and methods
10	Good and helpful comments

Assignment Submission Instructions

1. Answer Part 1 questions in a WORD, PDF, or any other document application

2. Compress the whole Eclipse source code of your project in part 2 in zip format using the Export function

Submit two items above to D2L.