Java Programming I: Exercise 7

The deadline for submissions is one week after the exercise.

The aim of this exercise is to learn the concept of polymorphism.

Source code of the lecture examples

- PolymorphismDemo.java
- PolymorphismDemo2.java

[Core Set, Problem 1] People's Introduction

(8 points)

Please study Example 1 (Lecture on Polymorphism, Part 1, Slides 7 and 8, PolymorphismDemo.java). You know that practically every language has special greeting words. People use them when meet each other for the first time. In English, it is: "How do you do?..." . In Japanese, it is: "Hajime mashite. ...". Your task is to create the hierarchy of classes to implement the different ways of greetings. On the top of the hierarchy should be the *Person* class. You should organize your hierarchy to get maximum benefit from inheritance and code reuse. See Table 1 for details.

To test your solution, you should create your data set (array of *Persons*) consisting of one person of each type and a demonstration program.

Your Java code should be saved to the *PeopleIntroduction.java* file.

Table 1. Ways of Greeting

Class	Greeting
Person	My name is...
Student	I am a student. My name is ... My ID is ...
EnglishPerson	How do you do? My name is ...
EnglishStudent	How do you do? I am a student. My name is ... My ID is ...
JapanesePerson	Hajime mashite. My name is ...
JapaneseStudent	Hajime mashite. I am a student. My name is … My ID is …

[Advanced Set, Problem 2] Fuel Consumption

(16 points)

Imagine, you are a software engineer at the GOOD taxi company. The high gasoline price affected taxi business. To make the business more profitable, your company restricted usage of air conditioners installed in the cars. According to old regulations, an air conditioner was always on during working hours. New regulations are established: Car drivers are instructed to switch air conditioners on when they drive passengers and switch them off when there are no passengers inside. Every day after finishing work, taxi drivers are requested to submit a

report. See an example below.

Table 2. Report

Time to begin work	Time to finish work	Total number of kilometers traveled by a car	Total number of hours traveled with passengers	Maker
7	18	200	8	Honda

The example above says that a driver was at work from 7 am till 6 pm. The car was carrying passengers for 8 hours. The rest of the time the driver was waiting for clients. The car covered a distance of 200 kilometers (this number includes the total distance with and without passengers). The taxi company has cars of several models. Car characteristics are presented in the following table.

Table 3. Car characteristics

Maker	Fuel economy (km/liter)	Air conditioner fuel economy (hour/liter)
Honda	14	10.5
Toyota	15.5	9.5 (it is always on)
Nissan	13	No air conditioner
Van (Nissan)	10.5	6

Your task is to design software calculating fuel consumption at the end of each day by all the cars. Your application has to answer the question: How much fuel does the company save when applying new regulations on air conditioner usage? To solve the problem, you should create a hierarchy of car classes.

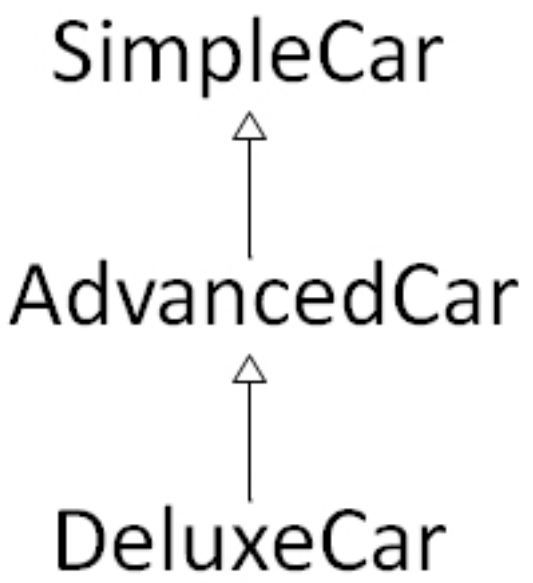

Fig. 1 Hierarchy of car classes

Cars without an air conditioner should belong to the *SimpleCar* class. Cars with an air conditioner which is always on should belong to the *AdvancedCar* class. The *DeluxeCar* class is for cars with an air conditioner which can be switched on and off by the driver. Every class should have methods with the following signature:

```
float calculateFuelConsumptionOldRegulations()
float calculateFuelConsumptionNewRegulations()
```

These methods should calculate the day fuel consumption taking into account the necessary parameters.

You should create a *FuelConsumptionCalculation* class:

```
class FuelConsumptionCalculation {
   private double fuelConsumptionOldRegulation = 0;
   private double fuelConsumptionNewRegulation = 0;
   private SimpleCar[] cars;

   FuelConsumptionCalculation() {
      /* your code is here */
   }
   FuelConsumptionCalculation(SimpleCar[] cars) {
      /* your code is here */
   }
   private void calculateFuelConsumption() {
      /* your code to calculate values of
        fuelConsumptionOldRegulation and
        fuelConsumptionNewRegulation is here */
   }
   float getFuelConsumptionOldRegulations() {
      /* your code is here */
   }
   float getFuelConsumptionNewRegulations() {
      /* your code is here */
   }
}
```

This class is to calculate daily fuel consumption of the taxi company cars. A default constructor should create an array consisting of one car of each type and calculate daily fuel consumption of the taxi company according to old and new regulations. The *FuelConsumptionCalculation(SimpleCar[] cars)* constructor takes an array of cars as a paratemeter. The *FuelConsumptionCalculation* may have other fields and methods. To test your solution, create your data set (array) consisting of one car of each type.

Your Java code should be saved to the *FuelConsumptionCalculation.java* file.