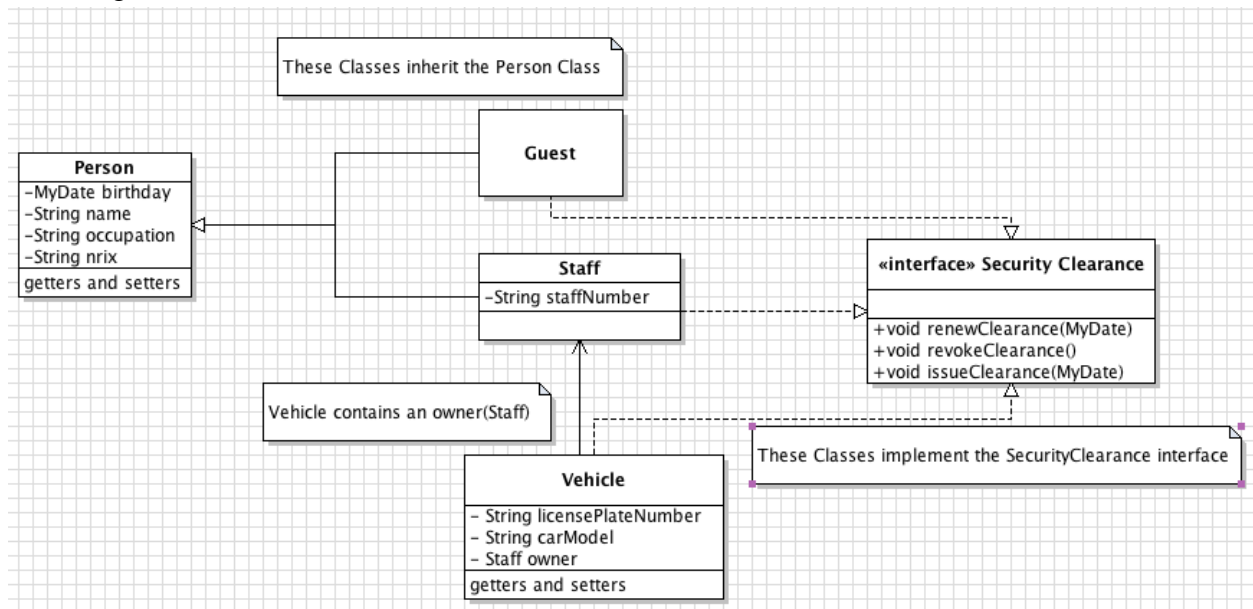


CS1020: Data Structures and Algorithms I

Tutorial 3 – Problem Solving with Java Library, String, and Java Generics (10th February 2012)

1. **[Polymorphism]** An important usage of polymorphism is for different objects that function differently to be called in the same context. This can be done through an interface. For example, a government agency with very high security keeps a list of guests, staff and vehicles that are allowed to enter its premises in its database, and you have been hired to program an infrastructure for them. The 3 types of clearance passes have different expiry dates and renewal rules.

You are given an almost complete infrastructure of the framework, which includes the following:



Person Class – This is the representation of a person, with some basic attributes. Guest and Staff classes inherit Person.

SecurityClearance interface – This is the interface that provides the security clearance to a person or a car. It should provide the following basic functions.

CS1020: Data Structures and Algorithms I

```
public interface SecurityClearance{

    //Checks if the clearance is valid at the given date

    //Changes the validity of the clearance

    public abstract void updateClearance(MyDate currentDate);

    //Issues a new clearance, the period of the clearance is based on the class that implements it

    public abstract void issueClearance(MyDate startDate);

    //revokes any present clearance

    public abstract void revokeClearance();

    //renews the clearance, the period of the renewal is based on the class that implements it.

    public abstract void renewClearance(MyDate startDate);

}
```

Staff Class – This is a child class of Person, and represents an employee, they enjoy a permanent clearance as long as they are employed. Their clearance function for them is revoked only when they leave the company, however, this should not be our concern as it is the job for the HR department, and our job is to ensure the function is provided.

Guest Class – This is a child class of Person. Guests are given 7 days guest pass each time and it is not renewed at the end of the 7 days.

Vehicle Class – This is a standalone class in Vehicle_Skeleton.java represents the vehicle of a Staff, and hence it has a Staff owner attribute.

However, it has yet to implement the SecurityClearance interface, and it is your job to do so. The rules of cars are as follows: their clearance lasts for one month, to the same day next month. For example, from 25th March to 25th April.

Renewal of the clearance requires a monthly fee, however, this is once again not our concern as it should be the job of Finance department. We are only concerned about providing all the necessary functionalities.

CS1020: Data Structures and Algorithms I

MyDate Class - This class is provided to help with date calculation. It has 3 attributes, which are the Days, Month and Year of a Date. As well as the following functions:

```
public class MyDate {  
    private int day;  
    private int month;  
    private int year;  
  
    //Start date should be earlier than end date.  
  
    //Will return negative date otherwise.  
  
    //Returns the number of days between the 2 dates.  
  
    public static int dateDifference(MyDate startDate, MyDate endDate);  
  
    //Returns the date one month later, eg. 12/1/2006 will yield 12/2/2006  
  
    public MyDate advMonth()  
  
}
```

Your Task: You are to read through the given infrastructure and complete the implementation of SecurityClearance Interface on the **Vehicle Class** by modifying the given Vehicle_skeleton.java code. This question aims to train you in understanding specifications and providing functionalities dictated by an interface. You should then test your class by running the SecurityDriver program.

Sample Output: SecurityDriver

```
$ java SecurityDriver
```

James Foo no longer has any clearance.

Peter Griffin has permanent clearance from 1/1/2012.

The vehicle owned by Peter Griffin, model AUDI RS3 no longer has clearance.

Lim Swee Teng has permanent clearance from 1/1/2012.

CS1020: Data Structures and Algorithms I

2. **[Multiple Inheritance]** Ever heard of multiple inheritance? No? That's because Java has explicitly disallowed multiple inheritance.
 - a. Explain what is multiple inheritance.
 - b. Find out why Java disallows multiple inheritance, and the solution Java has for it. (Please Google for the answer.)

3. **[Java String]** Strings are a very important part of any program that you write. It is important to know the different properties of Strings.
 - a. (Immutability) Strings are immutable. Can you think of any reason why? Is there a mutable version of String in Java? (Don't be lazy, Google for an answer!)

 - b. (Comparison) We have learnt how to use `.compareTo()` in the lecture. This method takes into account the different letter cases in the String. Is there a way to return equality disregarding letter cases? For example, comparing "iLoVeCs1020" and "ilovecs1020" should be equal. There are at least 3 ways to do this.

 - c. (Concatenation) Try the following in Java. What does it print? Why? How can you make it print "12345" instead?

```
int x = 1, y = 2;  
System.out.println(x + y + "345");
```

 - d. (Challenge) Write a program that asks for a String from the user. The String will only be one line long (no newline characters in it) with zero or more words (separated by one or many spaces in between each word), and may contain empty spaces at the start and/or end of the line. The program must output:
 - i. "Empty!" If the string does not contain any words. A String with only spaces is considered an empty String.
 - ii. The last word of that String otherwise.

For example: Inputs " " and "" should return "Empty!". Input "Hello world!" should return "world!". Input "Hi..." should return "Hi...". Input "CS1020 Teaching team ROCKS !" should return "!" (Note there is a space in "ROCKS !").

There are many ways to achieve this, and you will need to use a wide variety of methods in the String class to do it. Try to create the shortest and/or most efficient code possible!

CS1020: Data Structures and Algorithms I

4. [Java Generics] One of the most powerful concepts in Java is the idea of Generics. You have already seen one such implementation in the lecture, which is the Vector class. Today, we are going to further explore the *Pair* class that you have seen in the lecture.
- Another way to implement the Pair class is to store 2 *Object* data types, because all data types in Java are subclasses of Object. Why do you think the inventors of Java decided to create Generics instead of using Objects?
 - Now let's implement a *Triple* class. Internally, the Triple class should only use the Pair class to store its data. Show your implementation of the *Triple* class. Remember to implement similar accessor and mutator methods get/set First()/Second()/Third(). (Hint: A triple can be implemented as a pair of an element and a pair.)

Here is a program written for you to test whether you have implemented your Triple class correctly:

```
public static void main(String[] args) {
    // Triple stores name, number of modules read, and CAP
    Triple<String, Integer, Double> student =
        new Triple<String, Integer, Double>("Ang", 10, 3.14);

    System.out.println("Name: " + student.getFirst());
    int numRead = student.getSecond();
    System.out.println("Modules read: " + numRead);
    double cap = student.getThird();
    System.out.println("CAP: " + cap);

    // This student read a new module and got A!
    student.setThird((numRead*cap + 5.0) / (numRead + 1));
    student.setSecond(numRead + 1);

    System.out.println("Name: " + student.getFirst());
    System.out.println("Modules read: " + student.getSecond());
    System.out.println("CAP: " + student.getThird());
}
```