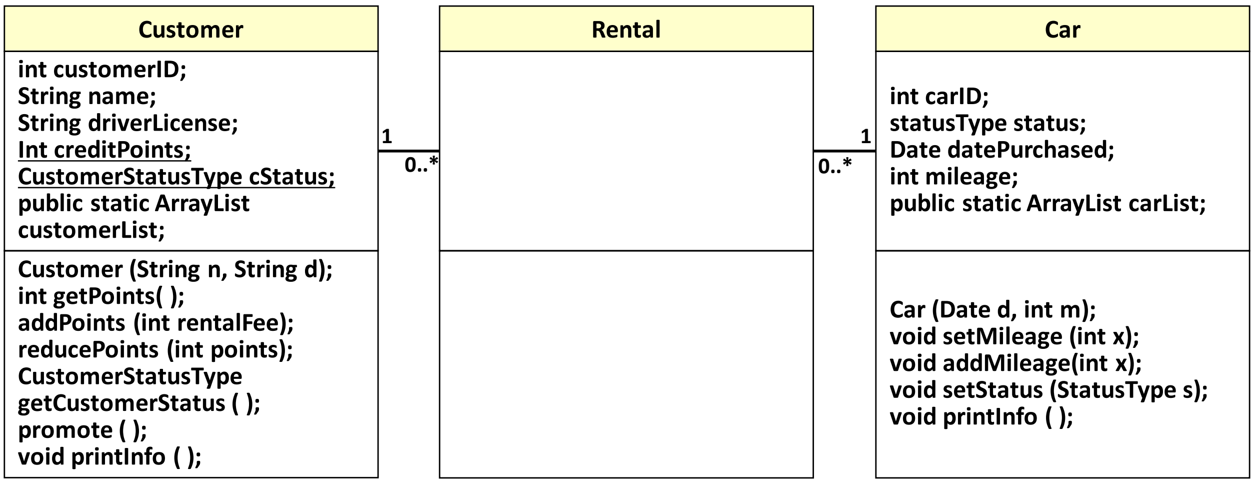
Laboratory 4. Static and Inter-class Relationships

(Due: Specified on Smart Campus web site.)

Spring 2022
Soo Dong Kim

1. **Write a Java program which implements the following classes and relationships.**



- Customer class
  - *CustomerID* is a unique 5-digit identification number for customers. The system generates this ID automatically using a random number generator inside the constructor. Check for a potential conflict with existing ID numbers.
  - *Name* is the name of the customer.
  - *Driverlicense* is the driver license of the customer.
  - *creditPoints* is the credit points earned from the customer' rental history. That is, customers earn credit points when returning cars. The point earning rates are slightly different for different levels of *cStatus*, as shown in the following table.

	Point Earning Rate
Silver	5%
Gold	10%
Diamond	20%

  - *CustomerStatusType* is an enumerated datatype, {*Silver, Gold*, *Diamond*}, representing the status of each customer.
  - *cStatus* represents the current status of the customer. initialize it as *silver* in the constructor.'
  - *customerList* is the list of all the customer instances created. Inside the constructor, each newly generated *Customer* instance is stored in this static array list.
  - *Customer( )* is the constructor that takes the parameters of the customer name and the driver license number. It assigns a customer ID and initializes the *creditPoints* with 0 and *cStatus* as Silver.

- *promote( )* method is to evaluate the rental history of the customer and change their *cStatus* the *total amount of rental fees paid*. For example, a customer with 300,000 wons for the total amount of rentals paid will be promoted to 'Gold' status.

	Minimum amount of Total Rental Fees
Silver	0
Gold	100,000
Diamond	500,000

  Since we do not implement Rental Class in this lab., the amount of total rental fees cannot be checked. Therefore, declare and use a local variable '*totalRentalFee*' in the *promote( )* method.
- printInfo( )

  This method is to print the information of the customer.

- Car class
  - *CarID* is a unique <u>4-digit number</u> identification number for cars, i.e., the range of 1,000 ~ 9,999." The system generates this ID automatically using a random number generator inside the constructor. Check for a potential conflict with existing ID numbers.
  - The system generates this ID automatically for each Car instance, using a random number generator. That is, Initialize the value of CarID inside the constructor. Check a potential conflict with existing CarID values. Upon a conflict, re-generate the carID.
  - *StatusType* is an enumerated datatype representing the status of each car. The datatype is defined with the values of (*available, checkedOut*, *inService*, *discarded*, and *sold*)." Initialize the value of *StatusType* with '*available'* inside the constructor.
  - *Mileage* is to represent the miles (or kilometers) that the car has been driven so far.
  - *carList* is the list of <u>all the car instances</u> created. Inside the constructor, each newly generated *Car* instance is stored in this static array list.
  - The 'd' and 'm' for the constructor are the initial values of *datePurchased* and *mileage* attributes respectively. Utilize *Date* and SimpleDate classes of Java library.
  - *Car( )* is the constructor that takes the parameters of the date purchased and the current mileage on the car. It assigns a Car ID and initializes the status with 'available'.
  - printInfo( )

    This method is to print the information of the car.

- Rental class

  This class is for the next assignment.

2. Define a main( ) method in *Rental* class that performs the followings;

- ❑ Creates 2 *Customer* instances using appropriate input parameter values.
- ❑ Print the information of all the customers.
- ❑ Creates 3 *Car* instances using appropriate input parameter values.
- ❑ Print the information of all the cars.
- ❑ Add the credit points of 300,000 to the customer #1 by using *addPoints( )*.
- ❑ Print the information of all customers.

3. Submission Guidelines

- ❑ Submit your solution on the web site; *myclass.ssu.ac.kr*
- ❑ Submit just **1 PDF file** containing the followings;
  - ❍ Java Source Code, *.java* file
  - ❍ Screenshot showing the program output
- ❑ Use this filename convention for your submission; **OOP.LAB.##.홍길동.pdf.**
  where ## is the laboratory number in 2 digits.
- ❑ No Plagiarism
  - ❍ The laboratory is an individual exercise. Do not copy others.
  - ❍ Submit your original work.

4. Grading Criteria (Total of 10 Points)

- ❑ Quality of Program (6)
  - ❍ Program Structure (4)
  - ❍ Exception Handling (1)
  - ❍ Header in the Source Program and Comments on Code (1)
- ❑ Accuracy of Output (4)
  - ❍ Correctness of Output Values (3)

    The output must be correct according to the problem specification.
  - ❍ Comprehensive Output Format (1)

    The output should be readable and comprehensive.
    Copy only the output part from the screen. (No Entire Screen)