Step 1: Registering Car, Owner and Insurance Coverage Details

All cars, owners and insurance coverage details will be registered into the system before processing any claim. The first line read from ***inputCICSData.txt*** contains three integers, which determine the number of registered cars, owners, and insurance coverage. For example, the total number of registered cars is (20), the total number of owners is (10), and the total number of insurance coverages is (10). In the following, we describe the format of each command.

1.1 Command: AddCar

This command is used to add all information of the registered cars. Car information includes plate number (such as WRF-5415), car type (such as car, CUV, pickup, bus etc), car brand (such as Toyota, Mercedes, Nissan etc), model name (such as Corolla, Civic, Fortuner etc), color and manufacture year. Check the following example and table.

Command Example
AddCar WRF-5415 Car KIA Optima Green 2021

Field name	Type	Example
Number Plate	String	WRF-5415
Type	String	Car
Brand	String	KIA
Model	String	Optima
Color	String	Green
Year	int	2021

1.2 Command: AddOwner

This command will add owner details into the system.

Command Example
AddOwner 2473823723 Tareq Abdali 1991 4 3

Field name	Type	Example
National ID	String	2473823723
First Name	String	Tareq
Last Name	String	Abdali

Date of Birth	Date	1991 4 3

1.3 Command: AddInsuranceCoverage

This command will add insurance coverage details into the system.

Command Example
AddInsuranceCoverage 101 Towing 150

Field name	Type	Example
Insurance Coverage Code	String	101
Insurance Coverage Description	String	Towing
Claim Amount	double	150

1.4 Command: Quit

The command quit will exit the process of entering the registration information.

Step 2: Process Insurance Claims

The claim details are provided in **inputClaims.*txt***. The first line of this file is an integer that determines the number of claims to be processed. For example, in the provided file ***inputClaims.txt***, 20 claims need to be processed. In the following, a more detailed description of processing insurance claims is explained.

2.1 Command: ProcessClaim

This command is used to process an insurance claim. This command contains the basic information of car, owner and insurance coverage. For each claim, it reads the insurance coverage code, car number plate, owner's National ID number, location (city name), date, insurance premium, and special offer.

Command Example
ProcessClaim 103 ORW-2024 2342114502 Jeddah 2020 5 28 false true

Field name	Type	Example
Insurance Coverage code	String	103
Number plate	String	ORW-2024

Owner's National ID	String	2342114502
Location	String	Jeddah
Date	Date	2020 5 28
Has Premium	Boolean	false
Has Special Offer	Boolean	true

Consider the following notes when processing claims:

Important Notes
• The system will read the ***Insurance Coverage Code*** as a string. You need to search for the insurance coverage object associated with the given insurance coverage code.
• The system will read the ***Number Plate*** as a string. You need to search for the car object associated with the given number plate.
• The system will read the ***Owner's National ID*** as a string. You need to search for the owner object associated with the given national ID.
• For each claim, the system should generate a unique 13-digit time stamped ***invoice number*** (Hint: Use System.currentTimeMillis() to generate it). Since it is a time stamped number, it will be different each time you run the program. This claim invoice number will be printed in the ***ClaimInvoices.txt*** (check step 3).
• **You must calculate total penalty amount** There are three types of fixed additional amount that will be added to the actual insurance coverage amount when applicable. 1. Premium Insurance Add additional amount of 200 SAR 2. Special Offer Add additional amount of 100 SAR 3. Senior Citizen Add additional amount of 50 SAR if owner age is 60 years or above (Use owner's date of birth to calculate age)

1.2 Command: Quit

The command quit will exit the process of entering the claim information.

Step 3: Print all the information

3.1 Print the registered car, owner, and insurance coverage information

As mentioned earlier, the registered car, owner, and insurance coverage information are read from **inputCICSData.txt** and are written to the file **CICSDatabase.txt.**

3.2 Print all issued tickets and report of total insurance coverages by each owner.

The system should print a log of all processed insurance claims. The system will calculate the total claim amount for each invoice. For example, in step 2, once the system processes the following command from the ***inputClaims.txt,*** the following information presented in the table below is written to the file ***ClaimInvoices.txt:***

Invoice details

```
Invoice No. 1645831569857

Insurance Coverage Details
Insurance Coverage Code: 103
Insurance Coverage Description: Color Damage
Insurance Coverage Penalty: 400.0

Car Details
Number Plate: ORW-2024
Type: SUV
Brand: Lexus
Model: RX470
Color: Grey
Built Year: 2017

Owner Details
National ID: 2342114502
Full Name: Jamal Albara

Claim Details
Date: 2020-28-5
Location: Jeddah

Total Amount: 600.0
```

After printing all claim invoices, the system will print a report showing a list of all owners and the number of claims by each owner as shown below:

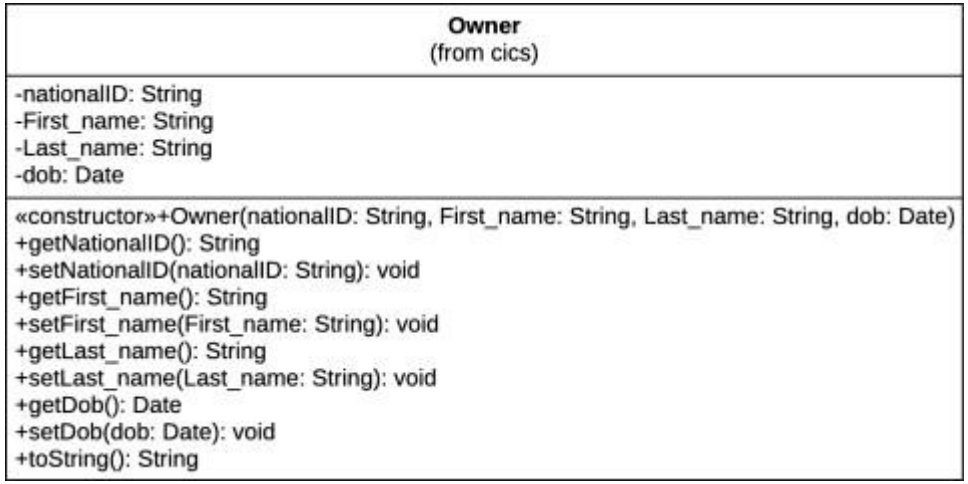
Number Of Claims by Owner

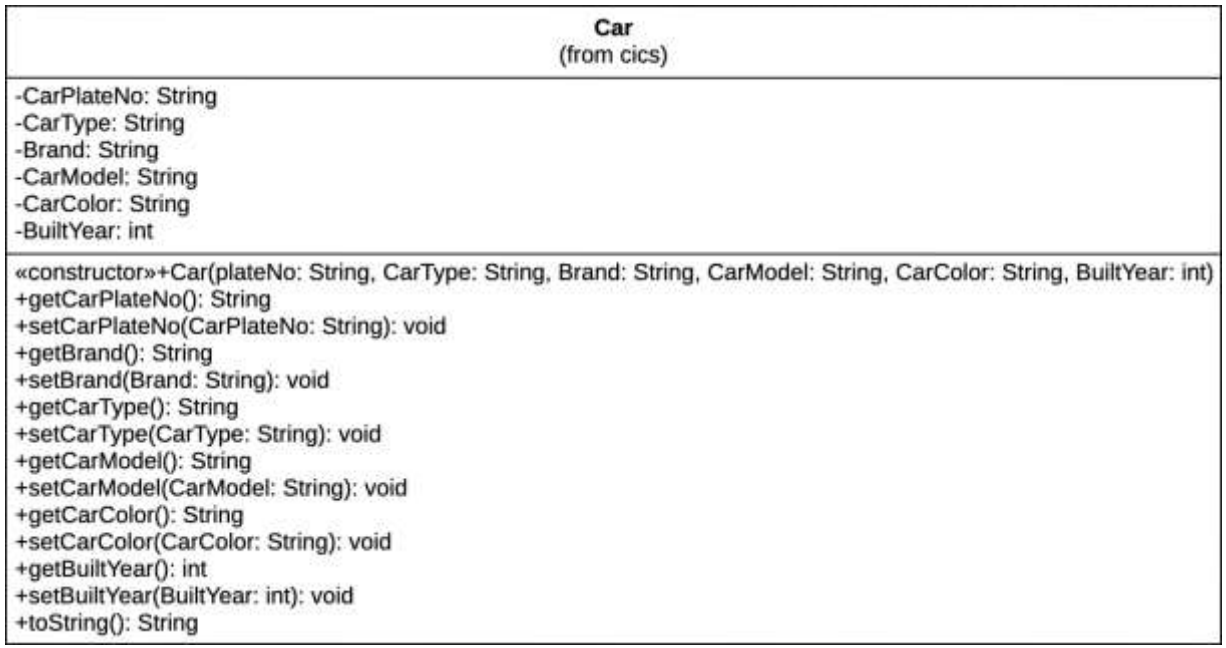
```
--------Total claim(s) by owner--------

Owner ID              Owner Name             Total Claim(s)
2473823723          Tareq Abdali                          3
2912367457           Bandar Omar                          3
2134367222           Khaled Bara                          0
2845898568          Hamid Farouk                          2
2021236524        Ahmed Shamrani                          0
2342114502          Jamal Albara                          3
2225198408          Abdul Rehman                          6
2474747448            Qasim Nasr                          1
2908967685          Sahl Zahrani                          0
2345347676          Abdul Rehman                          2
```

1.2 UML Class Diagram

In addition to the main class, you should create four classes as shown in the following UML diagram. Note that you should write appropriate constructor, setter, and getter methods for all classes. (You don't need to follow the same given arguments). Be aware of the visibility (publicprivate) for each attribute/method.


Owner
(from cics)
-nationalID: String
-First_name: String
-Last_name: String
-dob: Date
«constructor»+Owner(nationalID: String, First_name: String, Last_name: String, dob: Date)
+getNationalID(): String
+setNationalID(nationalID: String): void
+getFirst_name(): String
+setFirst_name(First_name: String): void
+getLast_name(): String
+setLast_name(Last_name: String): void
+getDob(): Date
+setDob(dob: Date): void
+toString(): String


Car
(from cics)
-CarPlateNo: String
-CarType: String
-Brand: String
-CarModel: String
-CarColor: String
-BuiltYear: int
«constructor»+Car(plateNo: String, CarType: String, Brand: String, CarModel: String, CarColor: String, BuiltYear: int)
+getCarPlateNo(): String
+setCarPlateNo(CarPlateNo: String): void
+getBrand(): String
+setBrand(Brand: String): void
+getCarType(): String
+setCarType(CarType: String): void
+getCarModel(): String
+setCarModel(CarModel: String): void
+getCarColor(): String
+setCarColor(CarColor: String): void
+getBuiltYear(): int
+setBuiltYear(BuiltYear: int): void
+toString(): String

InsuranceCoverage
(from cics)

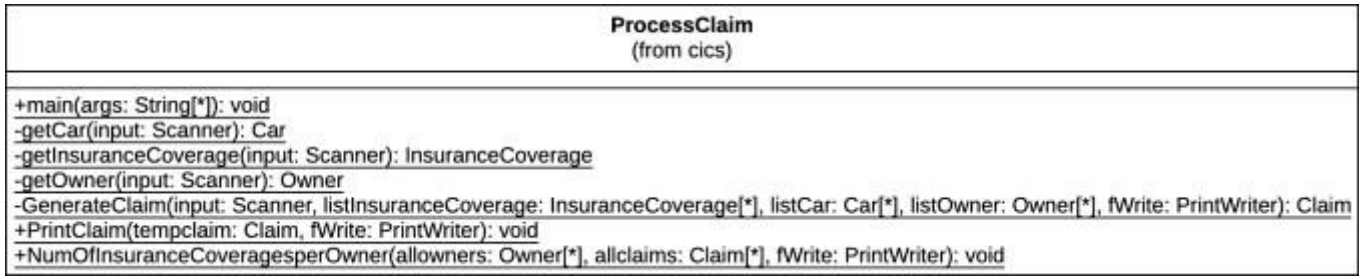
-insuranceCoverageCode: int
-description: String
-amount: double

«constructor»+InsuranceCoverage(insuranceCoverageCode: int, description: String, amount: double)
+getInsuranceCoverageCode(): int
+setInsuranceCoverageCode(insuranceCoverageCode: int): void
+getDescription(): String
+setDescription(description: String): void
+getAmount(): double
+setAmount(amount: double): void
+toString(): String

Claim
(from cics)

-ClaimNo: long
~location: String
~claimDate: Date
~hasPremium: Boolean
~hasSpecialOffer: Boolean

«constructor»+Claim(ClaimNo: long, insuranceCoverage: InsuranceCoverage, car: Car, owner: Owner, location: String, claimDate: Date, hasPremium: Boolean, hasSpecialOffer: Boolean)
+getClaimNo(): long
+setClaimNo(ClaimNo: long): void
+getInsuranceCoverage(): InsuranceCoverage
+setInsuranceCoverage(insuranceCoverage: InsuranceCoverage): void
+getCar(): Car
+setCar(car: Car): void
+getOwner(): Owner
+setOwner(owner: Owner): void
+getLocation(): String
+setLocation(location: String): void
+getClaimDate(): Date
+setClaimDate(claimDate: Date): void
+getHasPremiumInsurance(): Boolean
+setHasPremiumInsurance(hasPremium: Boolean): void
+getHasSpecialOffer(): Boolean
+setHasSpecialOffer(hasSpecialOffer: Boolean): void
+CalculateFinalClaimAmount(): double


ProcessClaim
(from cics)
+main(args: String[*]): void
-getCar(input: Scanner): Car
-getInsuranceCoverage(input: Scanner): InsuranceCoverage
-getOwner(input: Scanner): Owner
-GenerateClaim(input: Scanner, listInsuranceCoverage: InsuranceCoverage[*], listCar: Car[*], listOwner: Owner[*], fWrite: PrintWriter): Claim
+PrintClaim(tempclaim: Claim, fWrite: PrintWriter): void
+NumOfInsuranceCoveragesperOwner(allowners: Owner[*], allclaims: Claim[*], fWrite: PrintWriter): void