CSC8404 Assessed Courcework Car Rental Application

Interfaces

The interfaces and classes that can be used to create a Car Rental application are:

- **CarRentalSystem**
- **AbstractCarRentalSystem**
- **Drivable**
- **Car**
- **Person**
- **RegistrationNumber**
- **DrivingLicence**

Interface Description

- **CarRentalSystem Interface**

  This interface provides the required functionality needed for the implementation of a Car Rental application, since it allows the programmer or the user to interface with functions such as creating a rent contract, terminating a rent contract, getting a collection of the currently rented cars, getting the number of the available for rent cars, etc...

- **AbstractCarRentalSystem Abstract Class**

  This abstract class provides a skeletal implementation of the ***CarRentalSystem*** interface, thus minimizing the effort required to implement it, and providing a good starting point for the application creation process. Some of the simpler methods are implemented in this class, such as returning the collection of the currently rented cars, returning the total number of available cars for rent and the car currently being rented by a person. The other methods, more specific to each application, are left to be implemented by the application itself.

- **Drivable Interface**

  This interface describes the functionality that a car requires in this car rental system. It should be noted that any vehicle that is *drivable* and follows the functionality described by the ***Drivable*** interface can implement it, allowing for car rental applications that rent different types of vehicles.

- **Car Abstract Class**

  As with the ***AbstractCarRentalSystem*** class, this abstract class provides a skeletal implementation of the ***Drivable*** interface, and using it to implement the concrete types of cars is advisable, since most of the methods described by the interface don't need to be re-implemented, such as the method returning the vehicle's current tank capacity, or the method returning whether the tank is full or not.

- **Person Interface**

  This is a simple interface that describes a person or customer.

- **RegistrationNumber Abstract Class**

  This is an abstract class that describes a simple registration number, which is composed of string components. It provides access to all the components that a registration number may be composed of, and it provides access to the string representation of the registration number by overriding the "***toString()***" method.

- **DrivingLicence Abstract Class**

  Similarly to the ***RegistrationNumber*** class, this is an abstract class that describes a driving licence that is composed by a number of string components, has a date of issue, and can be either a full driving licence or not. It also provides access to each of these components.

  Both ***RegistrationNumber*** and ***DrivingLicence*** classes are abstract classes because they provide some default behaviour (such as guaranteeing that both these classes have unique representations) and using interfaces instead would move that default behaviour to the concrete classes, which as a result could lead to duplicate code.

Implementation

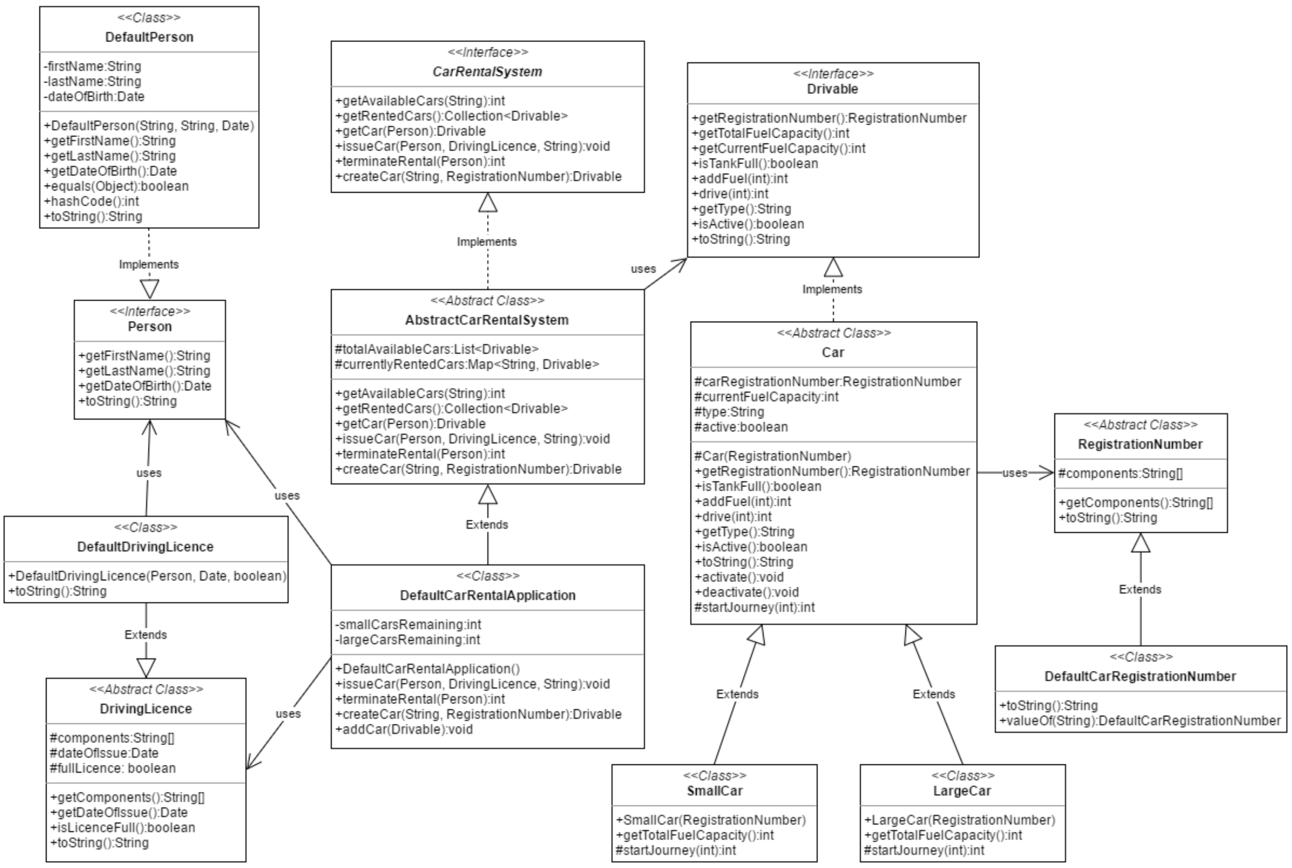
The following classes provide an concrete implementation of the aforementioned interfaces:

- **SmallCar**
- **LargeCar**
- **DefaultPerson**
- **DefaultCarRegistrationNumber**
- **DefaultDrivingLicence**

These classes, using the car rental interface described above, create the concrete classes used by the car rental system that is described in the System overview and Implementation sections of the coursework. They implement all the rules specified on those sections on how they should behave.

Finally, the **DefaultCarRentalApplication** class implements the car rental application that is described on those previously mentioned sections of the coursework and it is used to interface with the actual car system itself.

UML Class Diagram


<<Class>>
DefaultPerson
-firstName:String
-lastName:String
-dateOfBirth:Date
+DefaultPerson(String, String, Date)
+getFirstName():String
+getLastName():String
+getDateOfBirth():Date
+equals(Object):boolean
+hashCode():int
+toString():String
Implements
<<Interface>>
Person
+getFirstName():String
+getLastName():String
+getDateOfBirth():Date
+toString():String
uses
<<Class>>
DefaultDrivingLicence
+DefaultDrivingLicence(Person, Date, boolean)
+toString():String
Extends
<<Abstract Class>>
DrivingLicence
#components:String[]
#dateOfIssue:Date
#fullLicence: boolean
+getComponents():String[]
+getDateOfIssue():Date
+isLicenceFull():boolean
+toString():String
<<Interface>>
CarRentalSystem
+getAvailableCars(String):int
+getRentedCars():Collection<Drivable>
+getCar(Person):Drivable
+issueCar(Person, DrivingLicence, String):void
+terminateRental(Person):int
+createCar(String, RegistrationNumber):Drivable
Implements
<<Abstract Class>>
AbstractCarRentalSystem
#totalAvailableCars:List<Drivable>
#currentlyRentedCars:Map<String, Drivable>
+getAvailableCars(String):int
+getRentedCars():Collection<Drivable>
+getCar(Person):Drivable
+issueCar(Person, DrivingLicence, String):void
+terminateRental(Person):int
+createCar(String, RegistrationNumber):Drivable
uses
Extends
uses
<<Class>>
DefaultCarRentalApplication
-smallCarsRemaining:int
-largeCarsRemaining:int
+DefaultCarRentalApplication()
+issueCar(Person, DrivingLicence, String):void
+terminateRental(Person):int
+createCar(String, RegistrationNumber):Drivable
+addCar(Drivable):void
uses
<<Interface>>
Drivable
+getRegistrationNumber():RegistrationNumber
+getTotalFuelCapacity():int
+getCurrentFuelCapacity():int
+isTankFull():boolean
+addFuel(int):int
+drive(int):int
+getType():String
+isActive():boolean
+toString():String
Implements
<<Abstract Class>>
Car
#carRegistrationNumber:RegistrationNumber
#currentFuelCapacity:int
#type:String
#active:boolean
#Car(RegistrationNumber)
+getRegistrationNumber():RegistrationNumber
+isTankFull():boolean
+addFuel(int):int
+drive(int):int
+getType():String
+isActive():boolean
+toString():String
+activate():void
+deactivate():void
#startJourney(int):int
uses
<<Abstract Class>>
RegistrationNumber
#components:String[]
+getComponents():String[]
+toString():String
Extends
<<Class>>
DefaultCarRegistrationNumber
+toString():String
+valueOf(String):DefaultCarRegistrationNumber
Extends
Extends
<<Class>>
SmallCar
+SmallCar(RegistrationNumber)
+getTotalFuelCapacity():int
#startJourney(int):int
<<Class>>
LargeCar
+LargeCar(RegistrationNumber)
+getTotalFuelCapacity():int
#startJourney(int):int