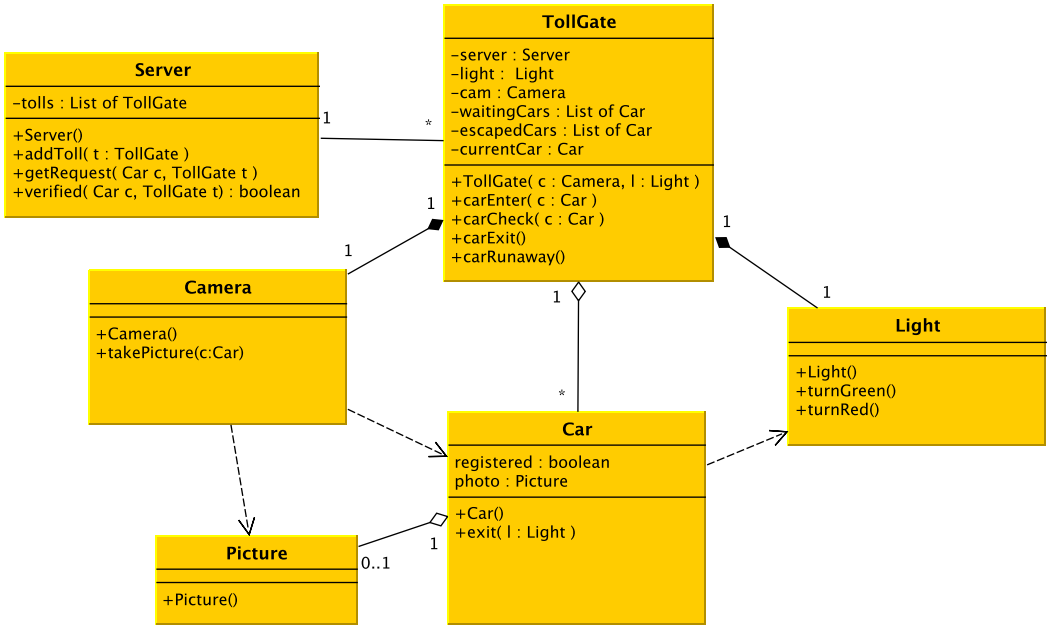
Seminar 5: Designing classes and interfaces
24292-Object Oriented Programming

1 Introduction

The aim of this seminar is to design a system to manage an Online Bookstore. More specifically the task is to identify and define the set of relevant classes, add all relations between each class, specify the cardinality and finally add attributes and methods for each class. Remember that the class diagram has to be a connected graph.
The solution to this exercise will be implemented in Java during the laboratory session 5.

1.1 A design example

Consider the following problem that we want to address, and design an application to solve it.
In most highways, tolls can charge cars automatically (for example via the Via-T system). We want to implement an application to manage this automatic charging procedure. To be able to participate, the driver has to register his car and then receives a device which has to be placed inside the car. The registration of the vehicle includes personal information of the driver (name, DNI, bank account number and car id). The automatic charging is done through a special lane of the toll where a sensor can sense the device in the car. The received information is used to charge the exact amount. When an authorized vehicle passes through the special lane a green light is turned on. If the vehicle is not authorized a photograph is made. The following picture shows a possible schematic (maybe incomplete) solution to the problem that you can discuss in class.



2 BookStore Manager

We want to implement an application which manages the books to sell of a virtual store with the following guideline. Note that the design should focus on managing the stocks and sales of books, and it is not necessary to model the store itself, nor how the users interact with it.

1. The store includes a catalog of books, and for each book we store its title, author, date, country, ISBN, price and currency.
2. When a user enters the web store a shopping cart is available which collects the books to buy: books can be added or deleted from it.
3. The user can acquire as many books as he wants of each copy.
4. The user can proceed whenever he wants to the payment of the items in the cart with a credit card.

A class that we will use is already defined, the Payment class which manages the checkout procedure of a user. This class is designed using the `Singleton` design pattern (that we will see in class) which ensures that only one instance of a class exists (is ever created). This is done by storing the instance of the class as a static attribute and adding a static getter method for accessing it `getTheInstance`. The constructor is made private so that it cannot be called to create more instances. A non-static public method exists `doPayment` to perform the payment with the given data.

```
<<Singleton>>
Payment
-$theInstance:  Payment
-Payment():  void
+$getTheInstance():  Payment {
   if (theInstance == null)
      theInstance = new Payment();
   return theInstance;
}
+doPayment(VISANumber:integer,ownerName:string,
           totalPrice:real,currency:Currency):  bool
```

In addition you must use the following classes from the package `java.util`:

- `java.util.HashSet`
- `java.util.Date`
- `java.util.Currency`

Before the seminar you are asked to look at the public API of those classes.