Software Engineering I

Object-Oriented Principles

Dr. Elham Mahmoudzadeh
Isfahan University of Technology
mahmoudzadeh@iut.ac.ir
2022

Object-oriented systems

- Focus on capturing the structure and behavior of information systems in little modules that encompass both data and process, called *objects*.
- A *class* is the general template we use to define and create specific instances, or objects.
- Every object is associated with a class.

Three parts of an object

- Attributes: describe information about the object.
- Behavior: specify what the object can do.
- State: defined by the value of its attributes and its relationships with other objects at a particular point in time.

Principles

- Abstraction
- Encapsulation
- Modularity
- Inheritance

Abstraction

- Is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.
- When you face with the problem, look at the most important things.
- Reduce the complexity to understand. Then, refine the problem and focus on the second level of the features.

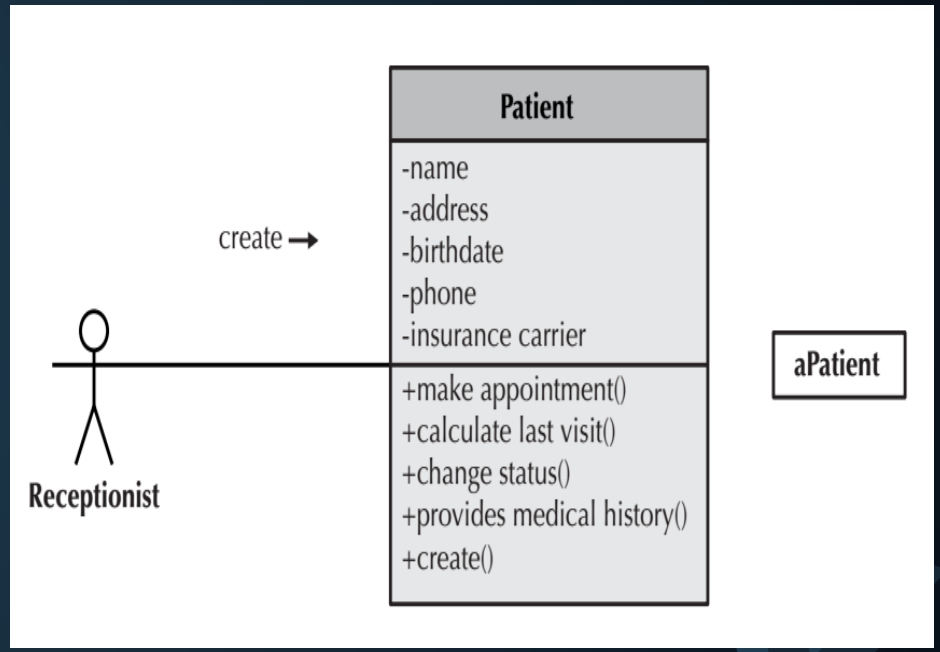
Encapsulation

- Is the combination of process and data into a single entity.
- See the class as a black box.
- Information required to be passed to the module and the information returned from the module are published.
- Exactly how the module implements the required functionality is not relevant. We really do not care how the object performs its functions, as long as the functions occur.
- It is used to hide the internal representation of an **object** from the outside.
- You cannot access to data of a class directly, but try to request to the class.
- Access level is very important.

Encapsulation(Cnt'd)

- The fact that we can use an object by calling methods is the key to reusability because it shields the internal workings of the object from changes in the outside system, and it keeps the system from being affected when changes are made to an object.

- The only information that an object needs to know is the set of operations, or methods, that other objects can perform and what messages need to be sent to trigger them.
  - *Messages* are information sent to objects to trigger methods.

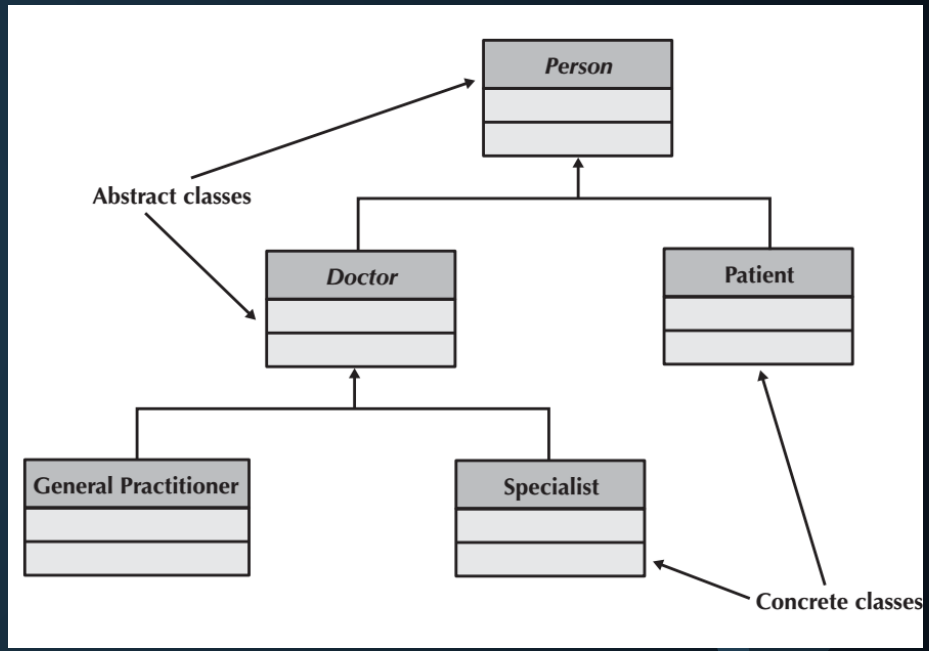


Modularity

- Modularity is the degree to which a system's components are made up of relatively independent components or parts which can be combined.
- Decompose a system into the objects that are loosely coupled with each other, connection should be as weak as possible.
- Objects should be highly cohesive, it is better for an object to be a single minded entity.

Inheritance

- Common sets of attributes and methods can be organized into *super classes*.
- Try to inherit data and operation from *superclass*.
- Makes it simpler to define classes. Instead of repeating the attributes and methods in the *subclasses*, the attributes and methods that are common are placed in *superclass* and inherited by the classes below it.



Benefits of Object-Oriented Systems Analysis and Design

- Concepts in the object-oriented approach enable analysts to break a complex system into smaller, more-manageable modules, work on the modules individually, and easily piece the modules back together to form an information system.

- The modularity makes systems development easier to grasp, easier to share among members of a project team, and easier to communicate to users, who are needed to provide requirements and confirm how well the system meets the requirements throughout the systems development process.

- By modularizing, the project, team actually is creating reusable pieces that can be plugged into other systems or used as starting points for other projects. This can save time because new projects don't have to start completely from scratch.

References

- Dennis, Wixon, Tegarden, “System Analysis and Design, An Object Oriented Approach with UML”, 5th Edition, 2015.

What we will talk about next...

- Object-Oriented approach