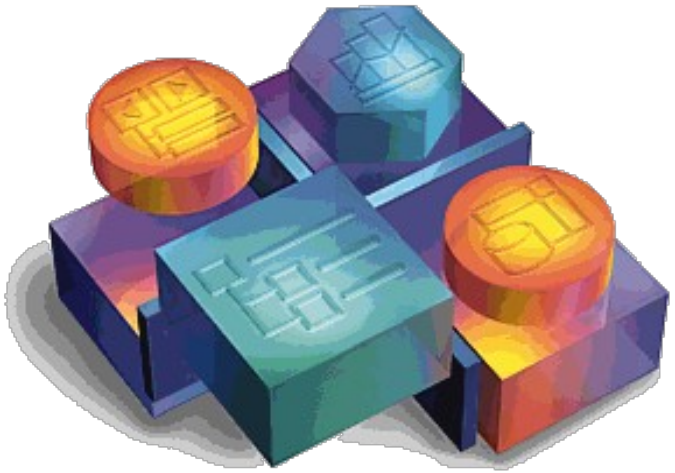
Object-Oriented Programming Design

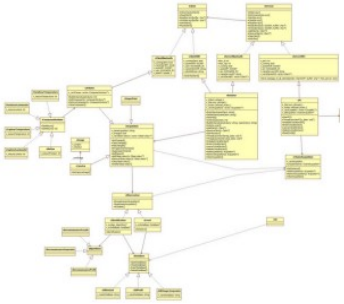


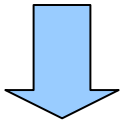
1. Introduction to Object-Oriented Concepts

2. How to think in Terms of Objects

3. Advanced Object-Oriented Concepts
4. The anatomy of a Class
5. Class Design guidelines
6. Designing with objects
7. Mastering Inheritance and Composition
8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns

2. How to think in Terms of Objects





► **Identify and solve business problems**

At first, don't consider a specific programming language.

Thinking about a robust and functional **object model**.

```
public class IntSquare {
        // private attribute
        private int squareValue ;
        //public interface
        public int getSquare(int value){
                SquareValue=calculateSquare(value);
                return squareValue;
        }
        //private implementation
        private int calculateSquare(int value){
return value*value;
        }
```

► **Code**

C++, C#, java...

2. How to think in Terms of Objects

- The fundamental unit of OO design is the **class**.
- The desired end result of OO design is a robust and functional **object model**.
- Many different ways to approach a problem.
- Be creative in solving problem.

2. How to think in Terms of Objects

Three important points to develop a good sens of OO thinking:

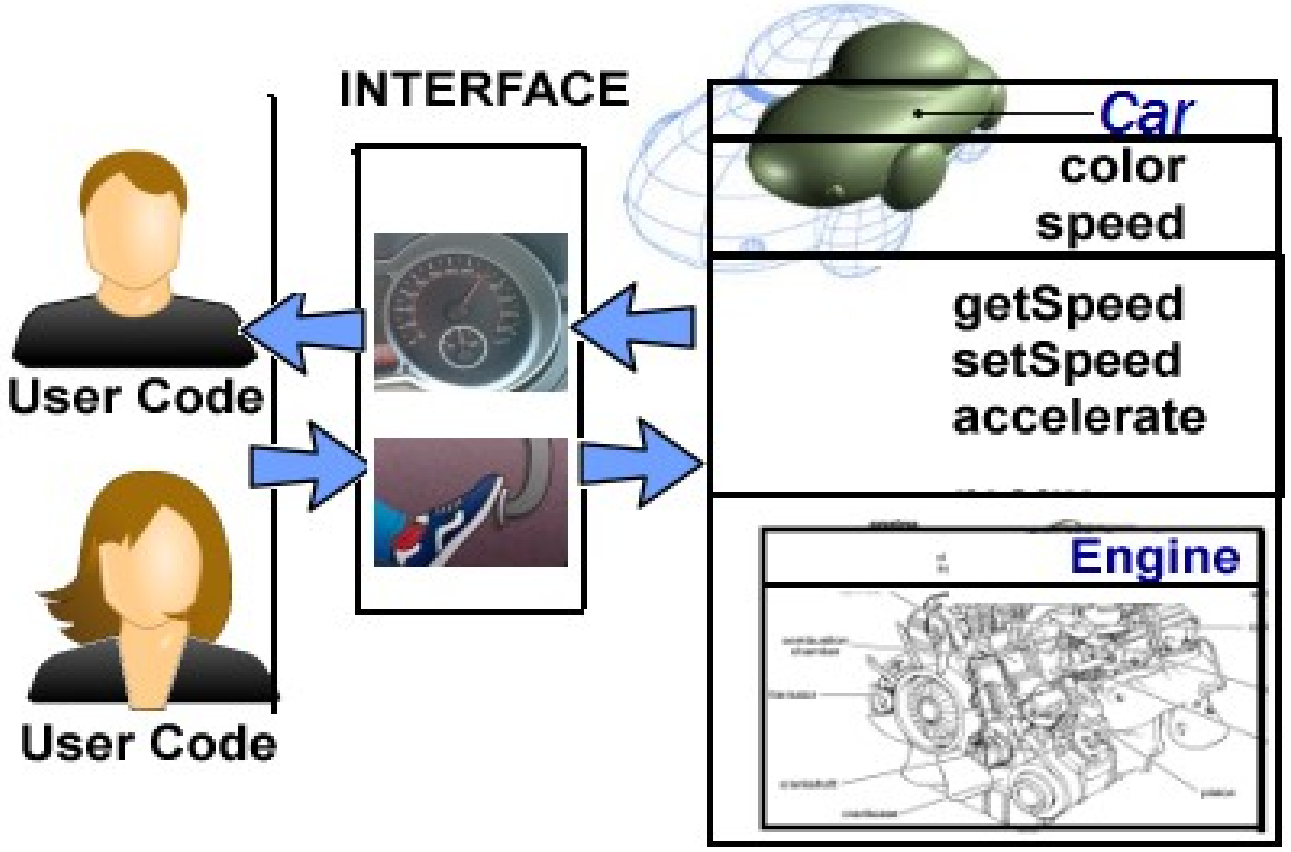
►Knowing the difference between the **interface** and **implementation**.

►Thinking more **abstractly**.

►Giving the user the minimal interface possible.

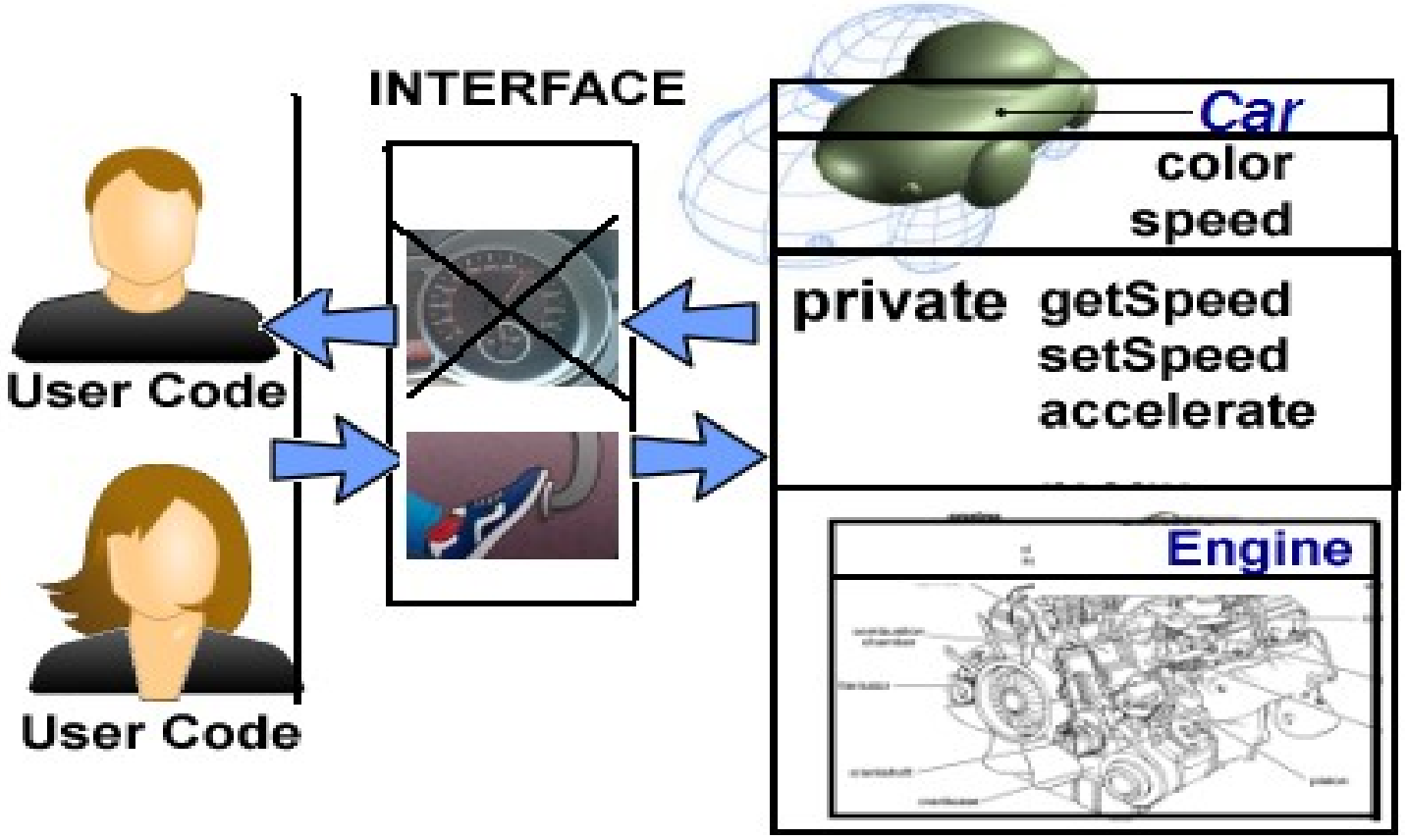
2. How to think in Terms of Objects

Knowing the difference between the interface and implementation.



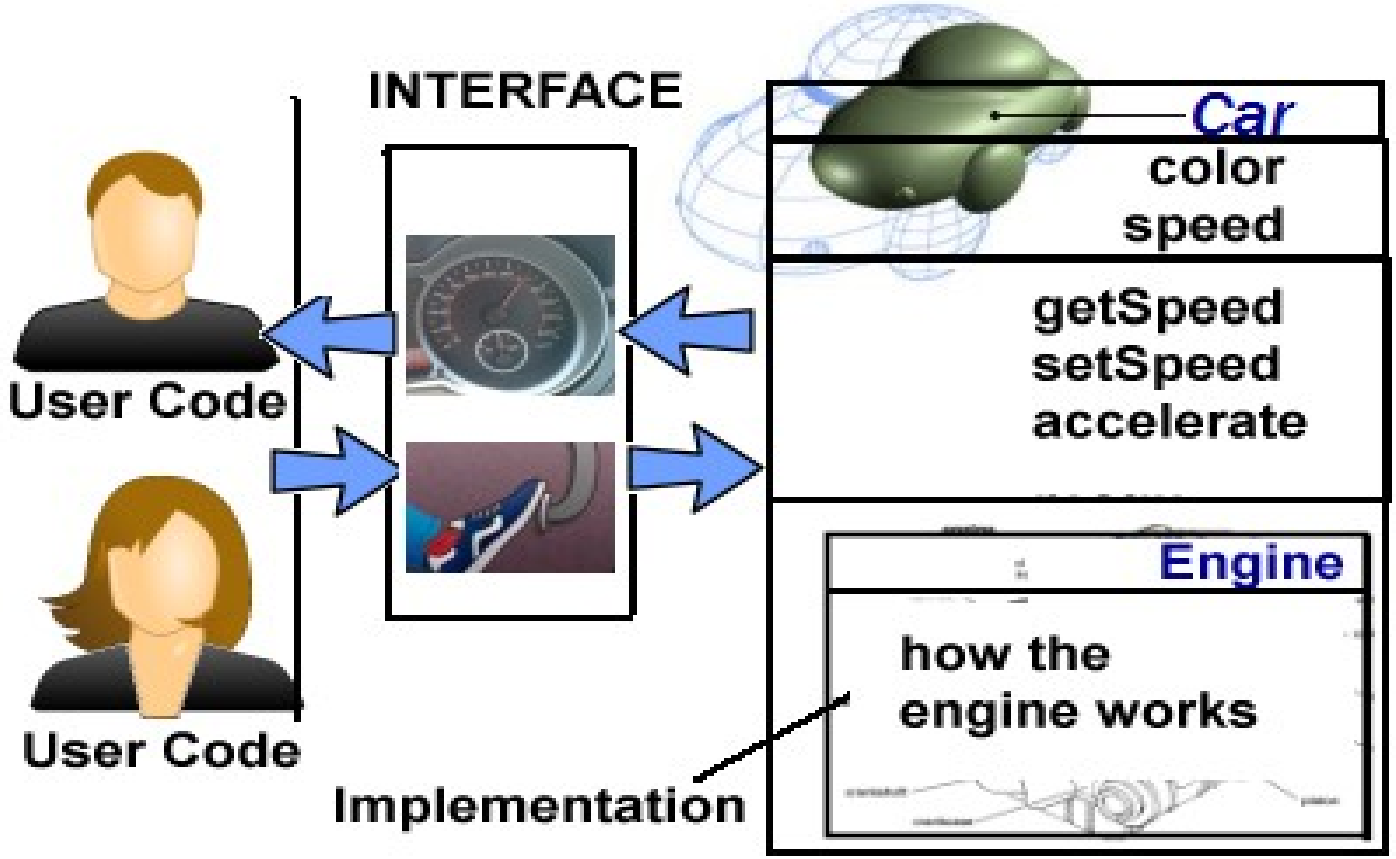
2. How to think in Terms of Objects

Knowing the difference between the interface and implementation.



2. How to think in Terms of Objects

Knowing the difference between the interface and implementation.



2. How to think in Terms of Objects

The interface:

getSpeed
setSpeed
accelerate

The services presented to an end user.

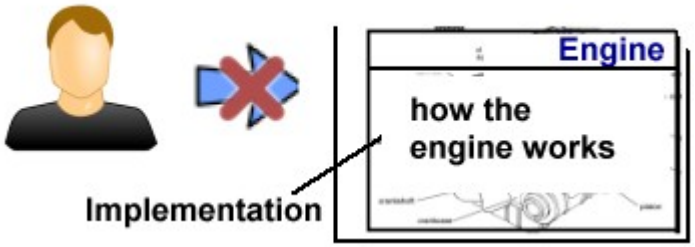
Think about the services the user needs.

►Identifying the user

The most inportant consideration when designing a class is identifying the audience, or users, of the class

2. How to think in Terms of Objects

The Implementation:



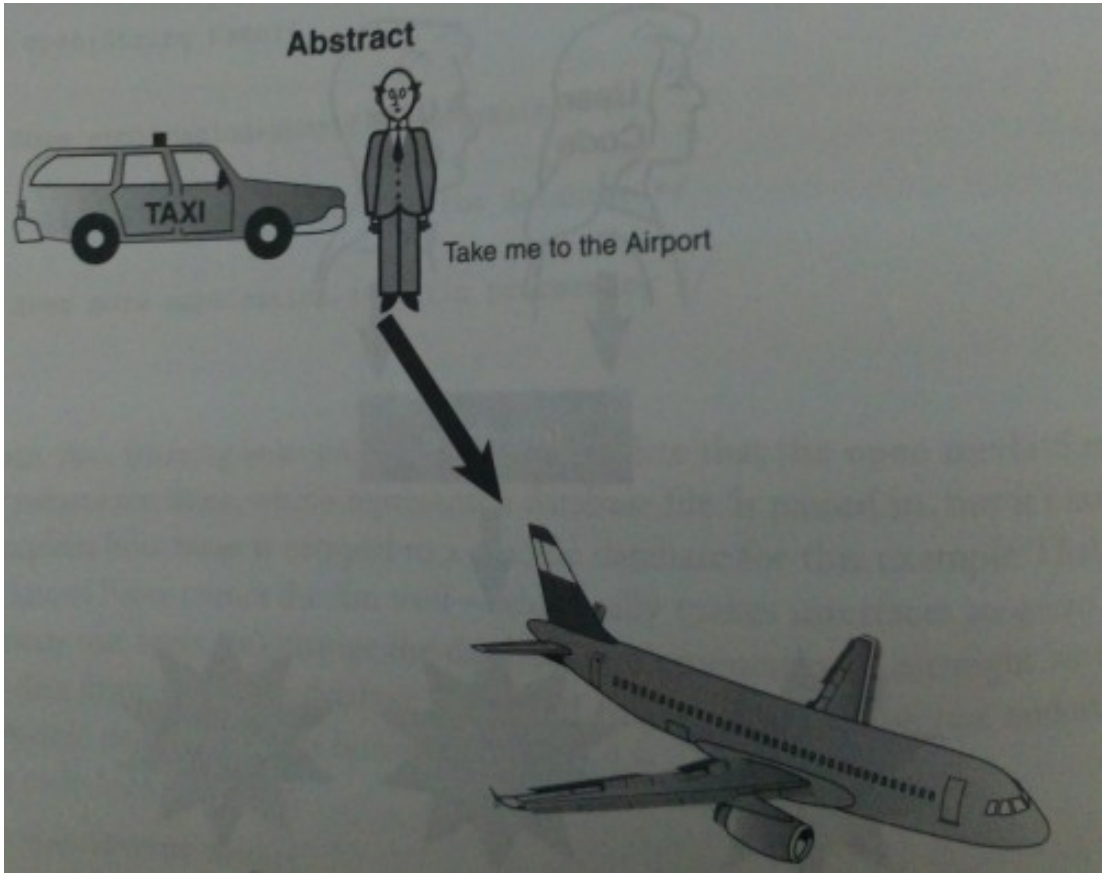
The implementation details are **hidden** from the user

A change to the implementation should not require a change to user's code.

For example: if we change the engine of the car, the user can continue to drive the car.

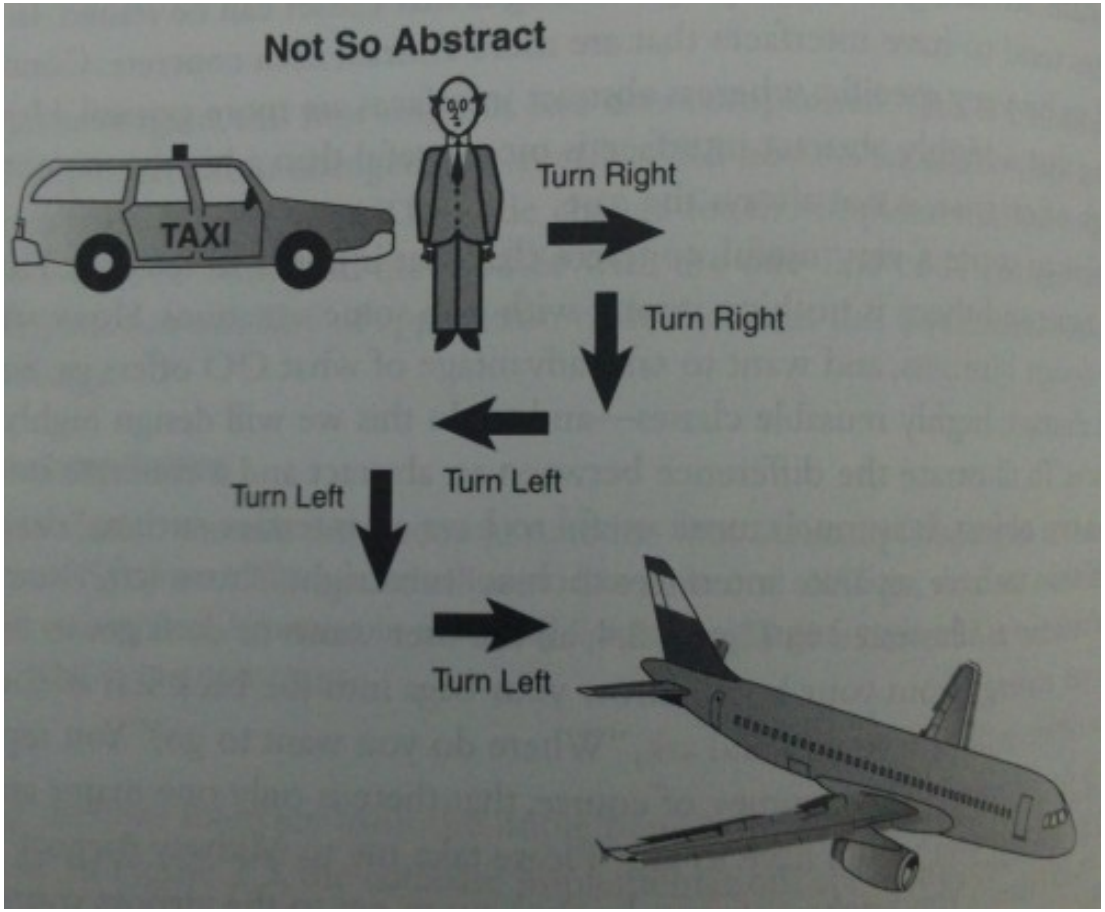
2. How to think in Terms of Objects

Abstract Interface



2. How to think in Terms of Objects

Concrete Interface



2. How to think in Terms of Objects

When designing a class, you have to think about:

►Give the users only what they absolutely need. Public interfaces define what the users can access.

►Design class from a user's perspective and not from an information system point of view.

►Make sure that the class will fit the requirements with people use it, not just developers.

2. How to think in Terms of Objects

Determining the Users:

►The users are the one who will actually use the system.

Any object that sends a message to the taxi object is considered a user. Users are objects too.

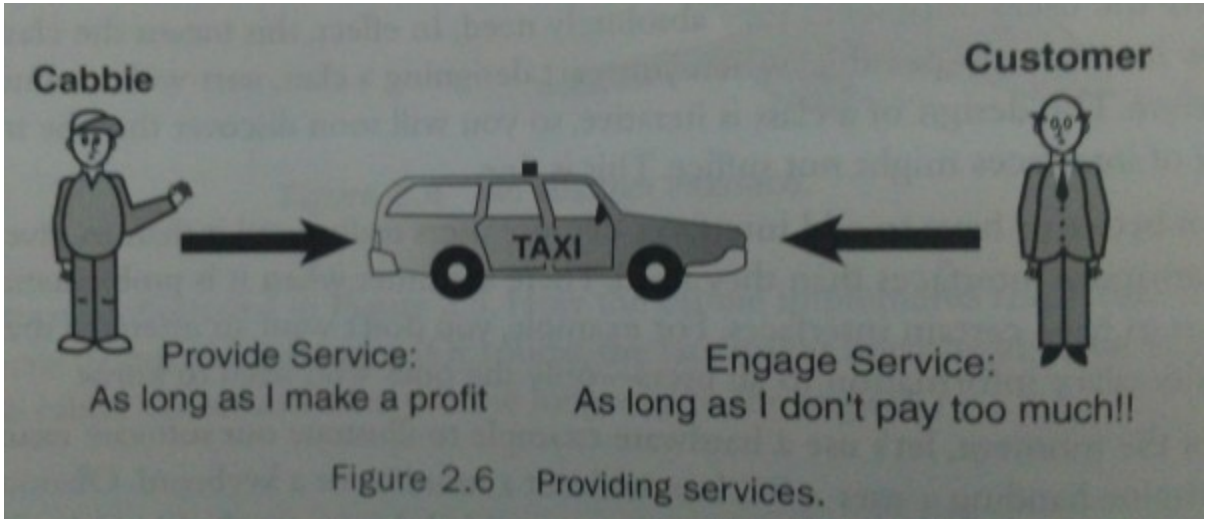


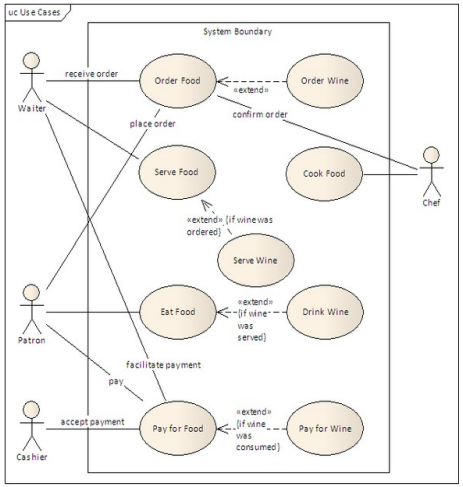
Figure 2.6 Providing services.

2. How to think in Terms of Objects

Object Behavior:

►After users are identified, you must determine the behaviors of the objects.

Using **UML UseCases** can help to identify behaviors of objects. Example of an UML UseCases:



2. How to think in Terms of Objects

Identify the Public Interfaces:

►Determine the public interfaces for each user object.

Example:

Think how you would use the taxi object:

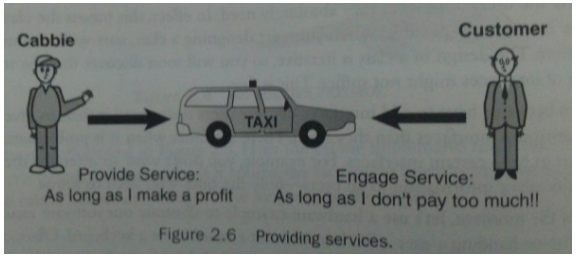


Figure 2.6 Providing services.

- Get into the taxi.
- Tell the cabbie where you want to go.
- Pay the cabbie.
- Give the cabbie a tip.
- Get out of the taxi.

What do you need to do to use the taxi object?

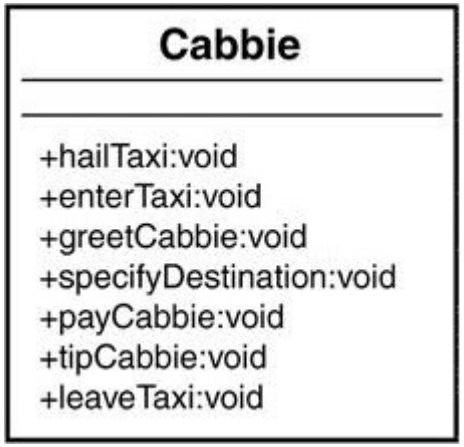
- Have a place to go.
- Hail a taxi.
- Pay the cabbie money

2. How to think in Terms of Objects

Identify the Public Interfaces:

You might discover that the object needs more interfaces,
such as «*Put luggage in the truck* » …

Nailing down the final interface is an **iterative process**.
Usually we have for each interface model only one behavior.



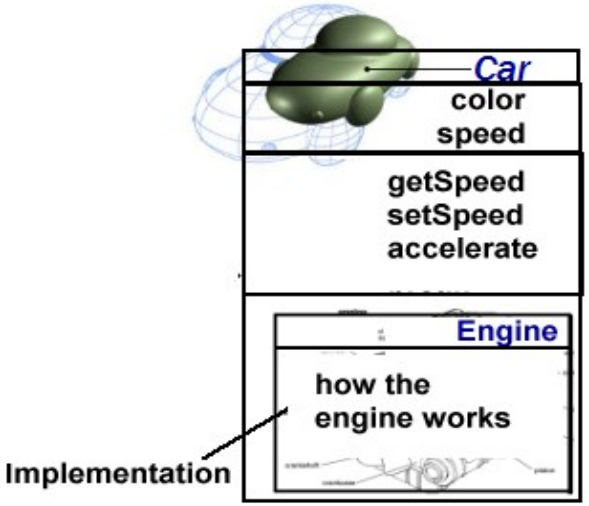
2. How to think in Terms of Objects

Identify the Implementation:

After the public interfaces are chosen, you need to identify the implementation.

- **Technically, anything that is not a public interface can be considered the implementation.**
- **Any private method is considered part of the implementation.**

  So the user will never see it.



2. How to think in Terms of Objects

Steps to design a Class:

- **1. Determining the Users**
- **2. Determining the behaviors**
- **3. Identifying the Public Interfaces**
- **4. Identifying the Implementation**

1. Introduction to Object-Oriented Concepts
2. How to think in Terms of Objects

3. Advanced Object-Oriented Concepts

4. The anatomy of a Class
5. Class Design guidelines
6. Designing with objects
7. Mastering Inheritance and Composition
8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns