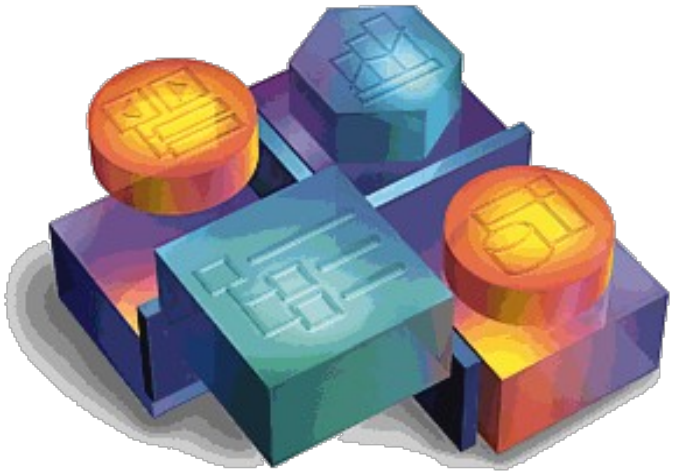
Julian 朱立安

QQ: 1603348107

1st Semester:
Object-Oriented Programming Design

2nd Semester:
Java Programming Design

Object-Oriented Programming Design



Name:

Class:

What do you want to do in the future? Have you some career plan?

Give me some details about (just few words):

OOP:

UML:

XML:

DTD:

XSD:

A 3-tiered system:

Thank you!

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

Different Programming Paradigms

Different ways to make a program...

Functional/procedural programming

program is a list of instructions to the computer

(Basic, C, Pascal, ...)

```
Begin
    if ... then
        first do this
        after do that
    else
        do this
end
```

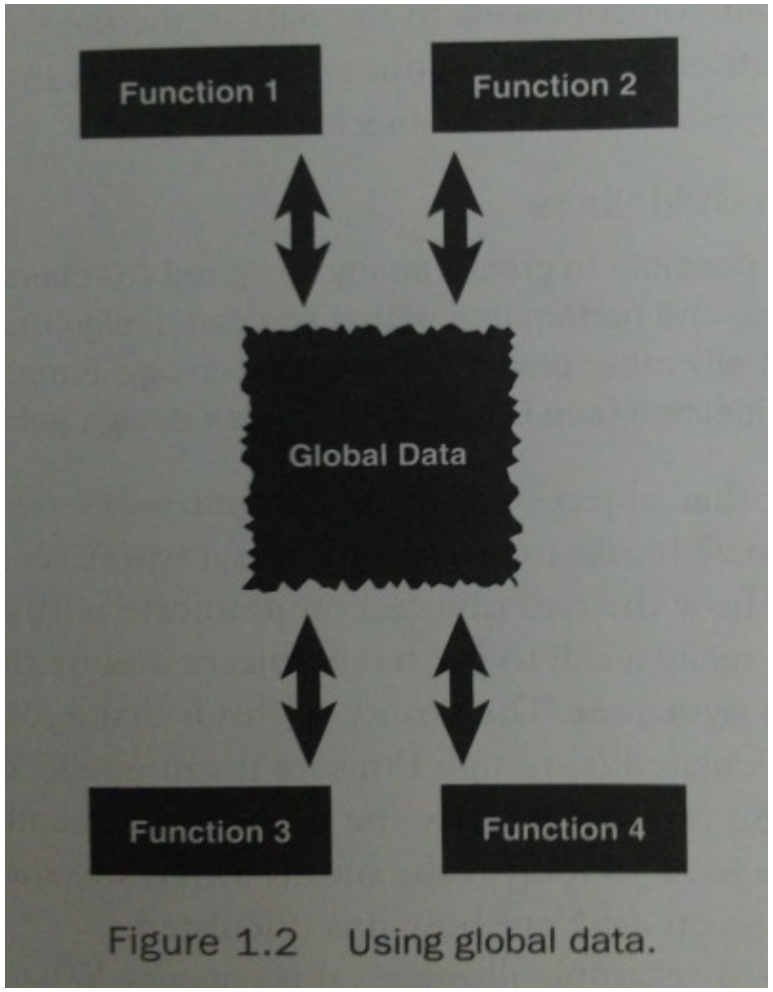


Figure 1.2 Using global data.

Different Programming Paradigms

Different ways to make a program...

Object-oriented programming

program is composed of a collection objects that communicate with each other

(Smalltalk, Java, C#, ...)

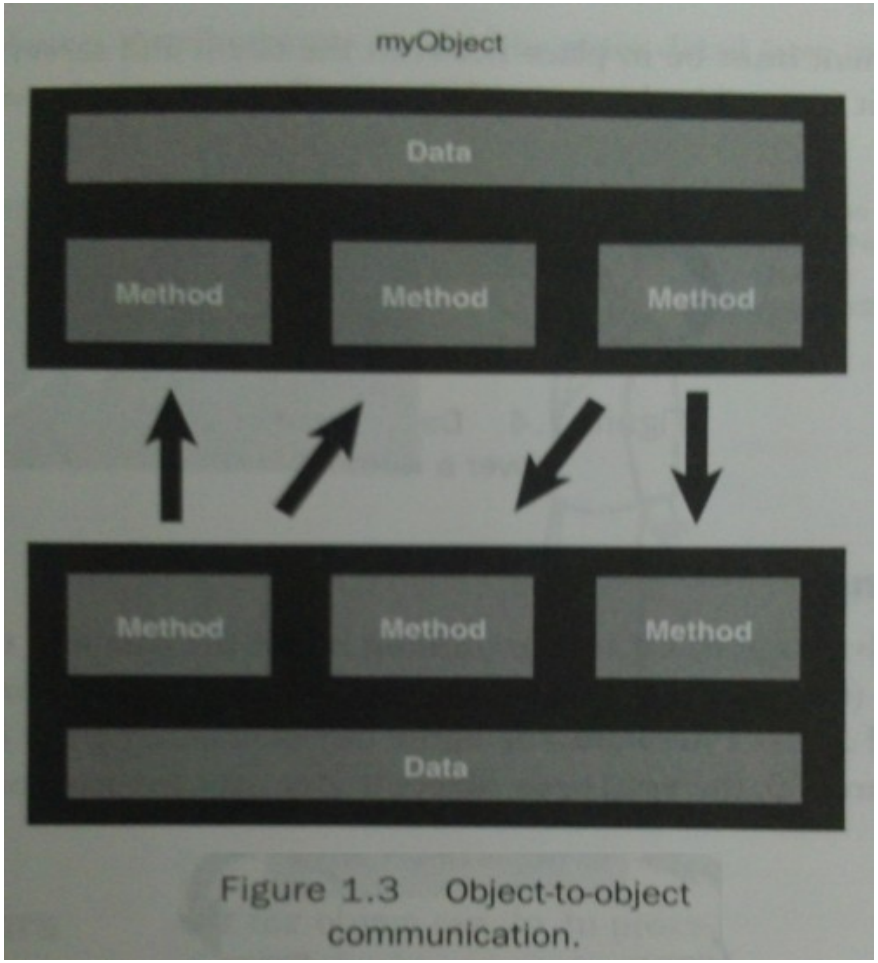


Figure 1.3 Object-to-object communication.

What exactly is an object?

- **Identity** – unique identification of an object
- **Object Data:** attributes – data/state

The data stored within an objects represents the state of the object. In OO programming terminology, this data is called **attributes.**

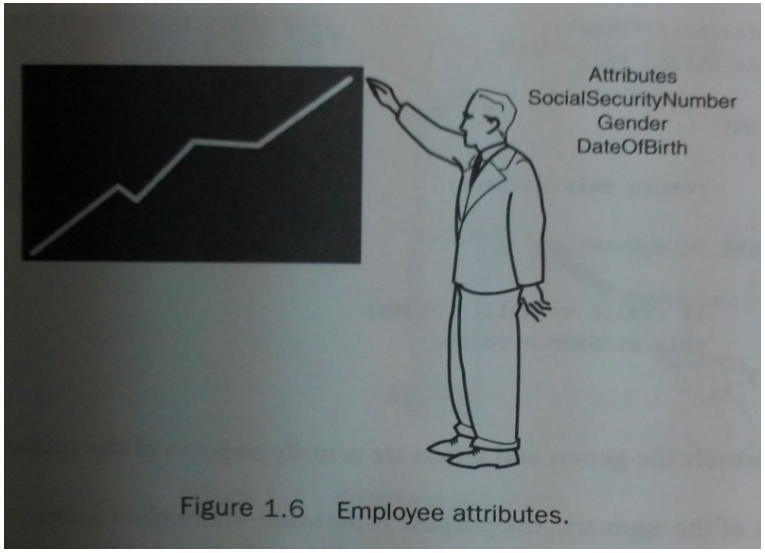


Figure 1.6 Employee attributes.

What exactly is an object?

- **Object Behaviors:** services – methods/operations
  - supported by the object
  - within objects responsibility to provide these services to other clients

  The behavior of an object is what the object can do. In OO terminology these behaviors are contained in methods.

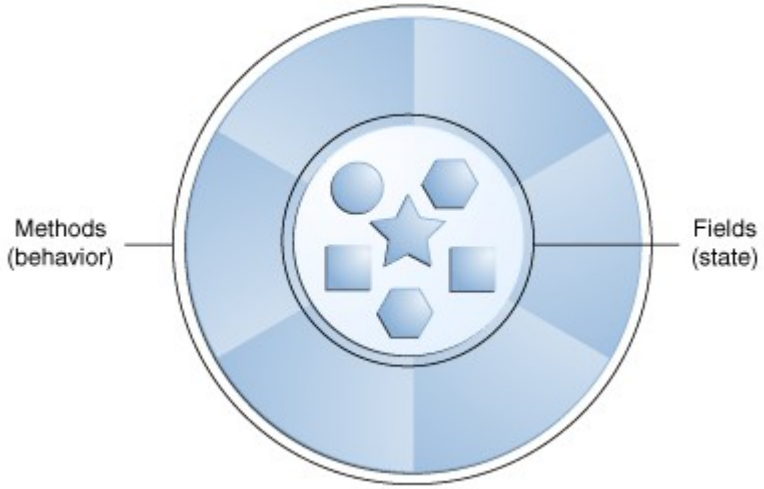
What exactly is an object?

- **Object Behaviors:**

Getters and Setters (accessor methods / mutator methods)
Providing controlled access to an object's data (Data hiding) .

- The name of the method
- The parameters passed to the method
- The return type of the method

```
public String getName() {
        return name;
}
```



What exactly is an object?

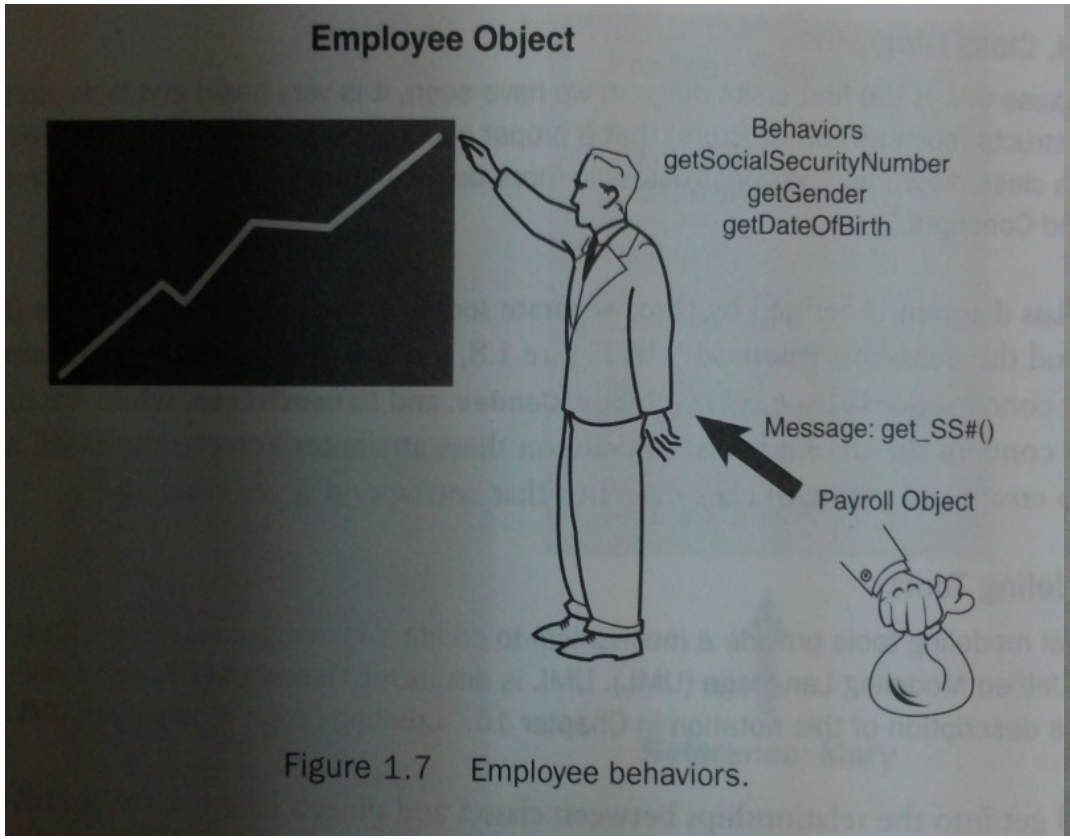
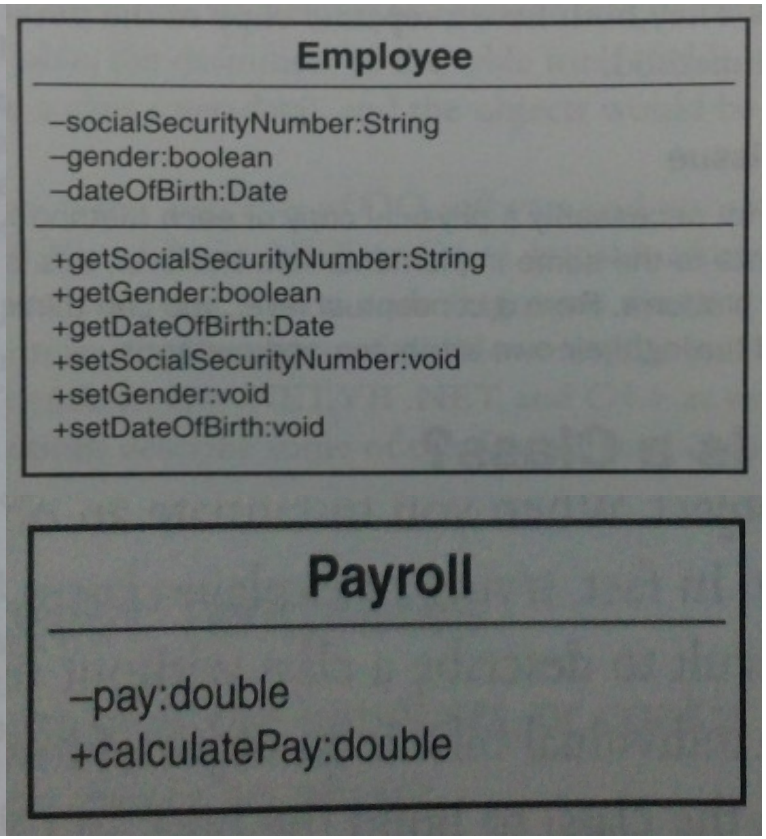
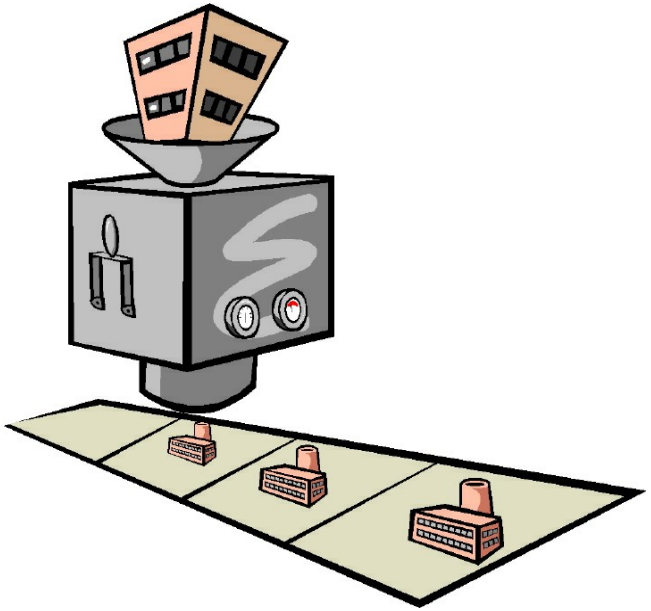


Figure 1.7 Employee behaviors.



What exactly is a Class?

- “type”
- object is an **instance** of class
- class groups similar objects
  - same (structure of) attributes
  - same services



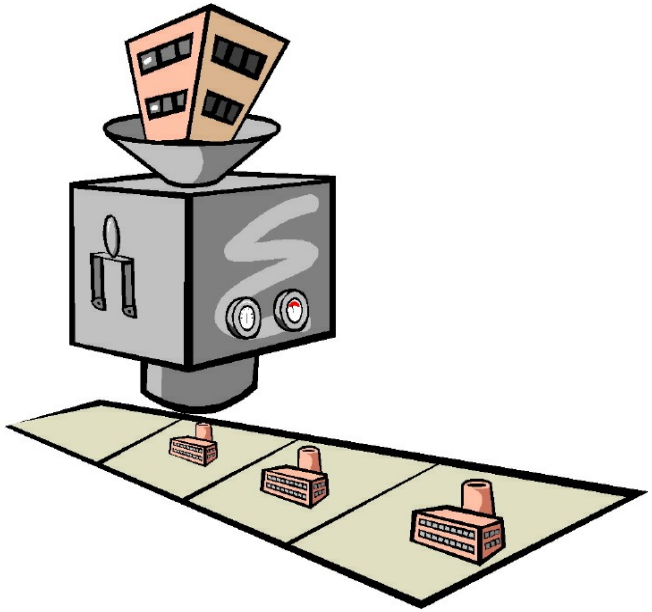
When you instantiate an object, you use a class as a basis for how the object is built.

What exactly is a Class?

- **Classes are Object templates**

Definition of a Person class

```
public class Person {
        // Attributes
        private String name;
        ...
        // Methods
        public String getName() {
                return name;
        }
        ...
}
```



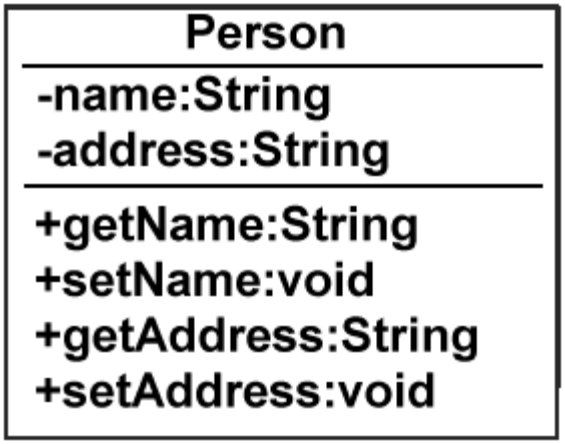
public: other object can directly access it

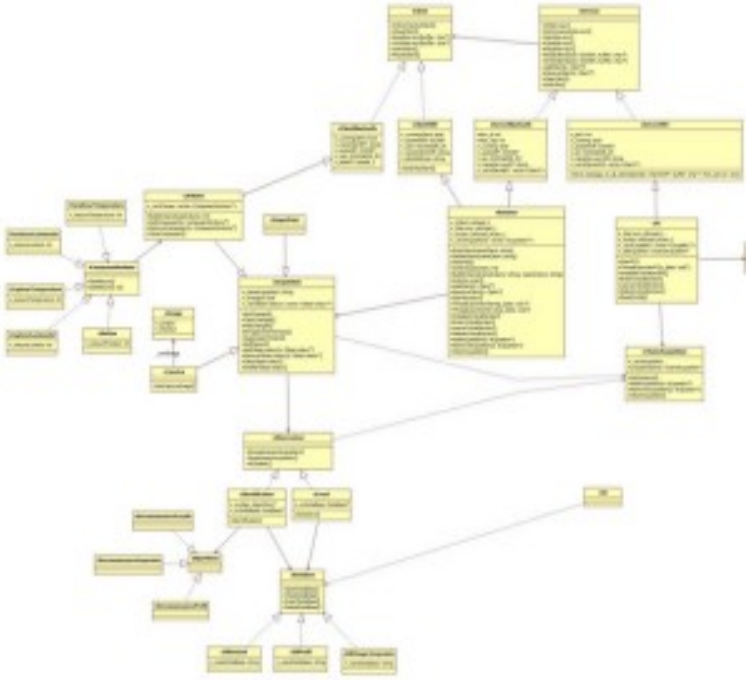
private: only that specific object can access it

Using UML to Model a Class Diagram

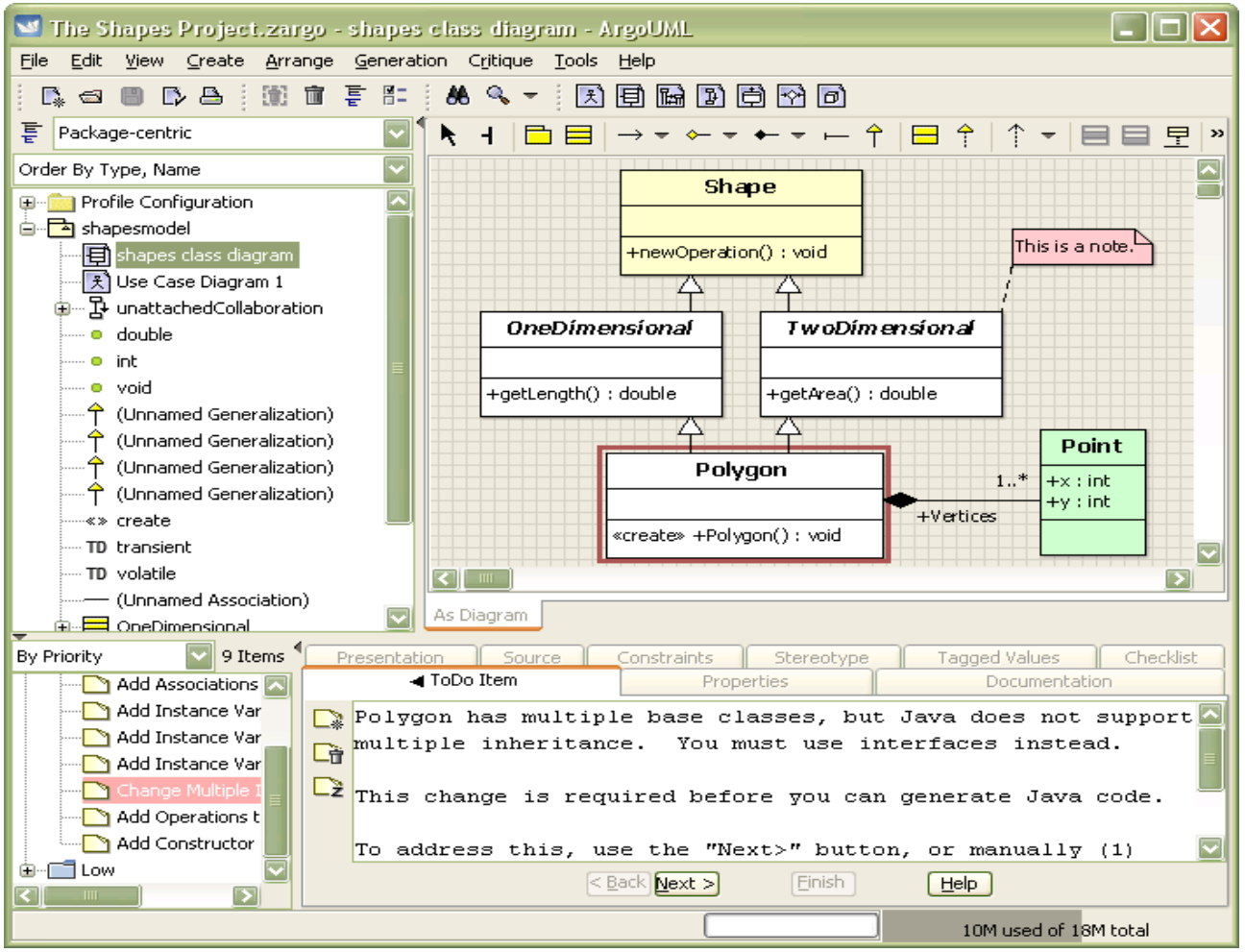
One of the most popular tools in use today is

UML: Unified Modeling Language





Using UML to Model a Class Diagram



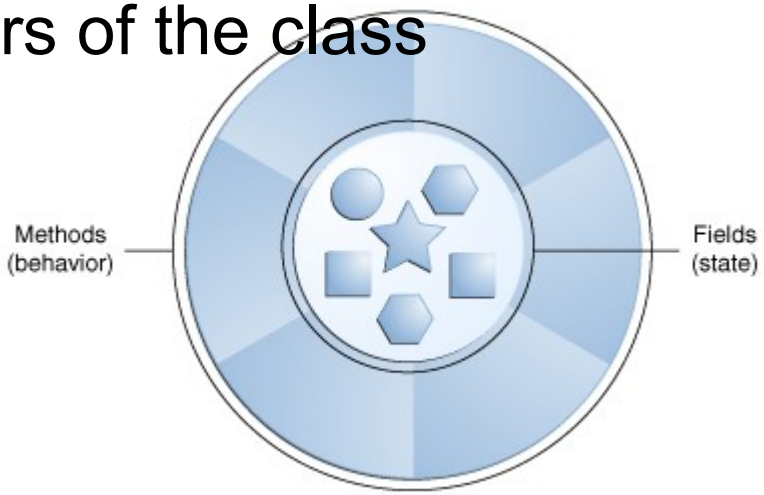
http://argouml.tigris.org/

Encapsulation and Data Hiding

- Separation between internal state of the object and its external aspects

  An object should only reveal the interfaces that other objects must have to interact with it.

- How ?
  - control access to members of the class



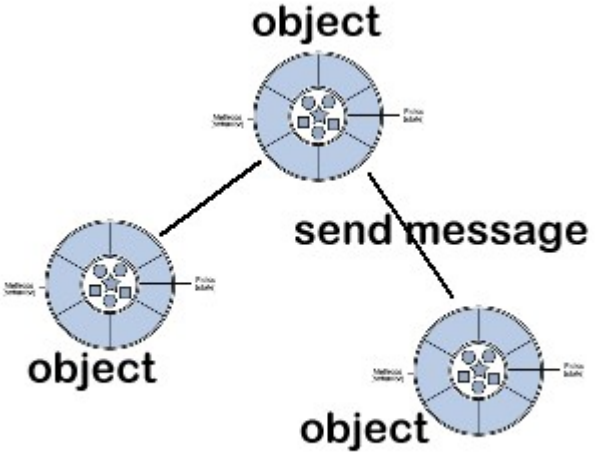
Encapsulation and Data Hiding

- **Interfaces:**

It defines the communication between objects.

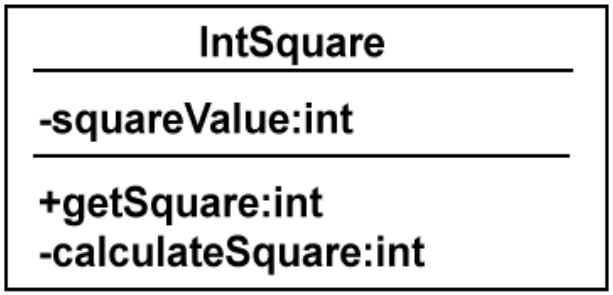
The interface describes how users of the class interact with the class.

In Most OO languages, the methods that are part of the interface as designed as **public**.



Encapsulation and Data Hiding

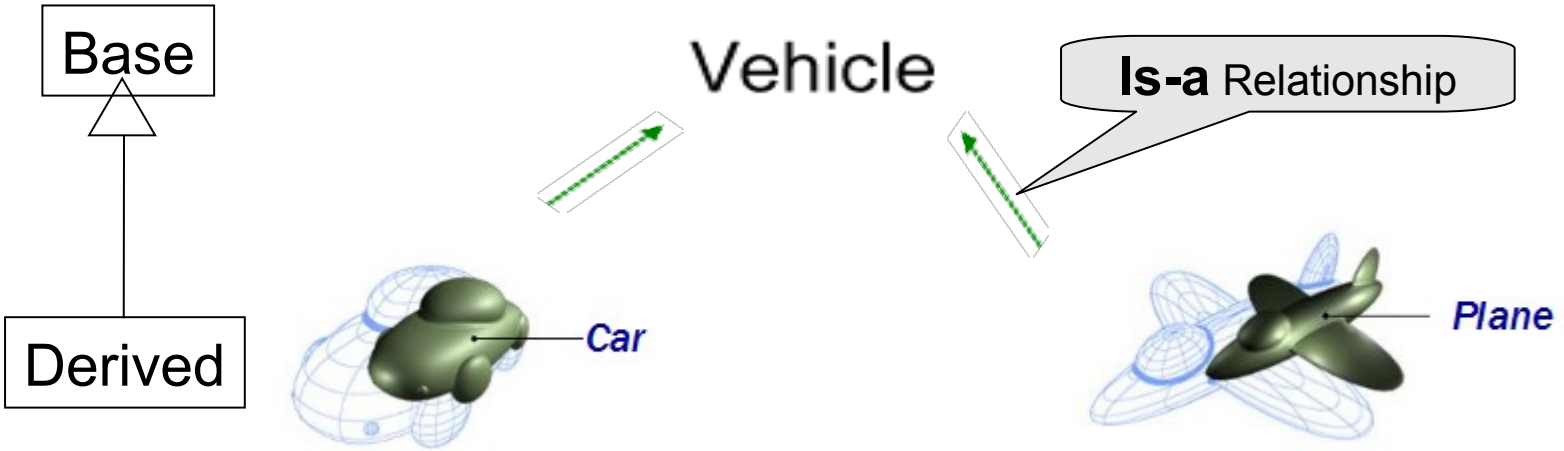
- **A model of the interface/Implementation Paradigm**




```
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;
        }
}
```

Inheritance

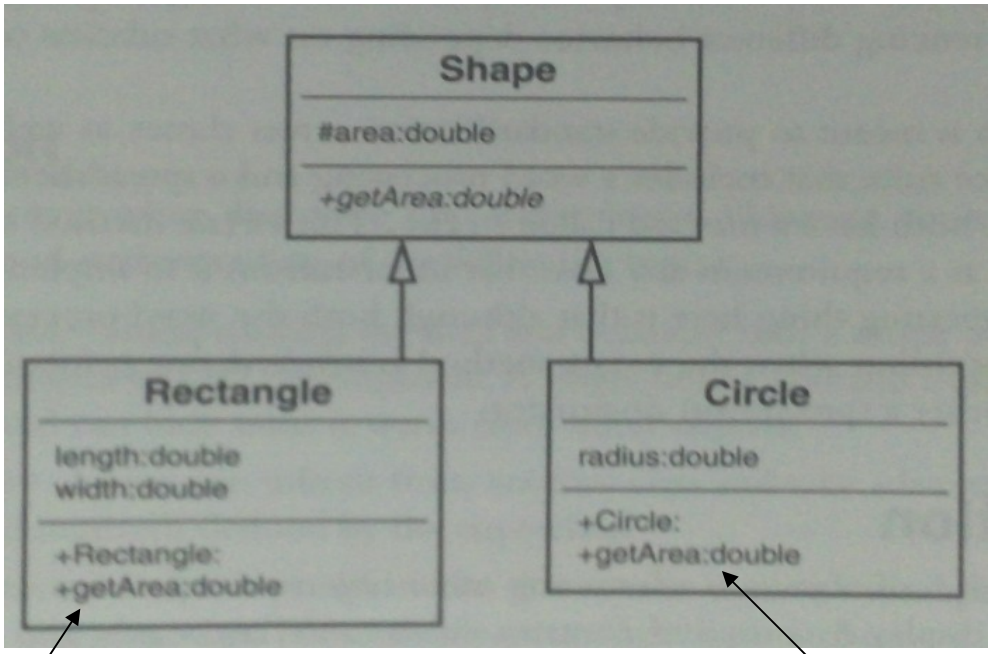
- Class hierarchy



- Generalization and Specialization
  - subclass inherits attributes and services from its superclass
  - subclass may add new attributes and services
  - subclass may reuse the code in the superclass
  - subclasses provide specialized behaviors

Polymorphism

- **Polymorphism means that similar objects can respond to the same message in different ways.**




```
public double getArea(){
   area=3.14*(radius*radius)
   return (area);
}
```

```
public double getArea() {
   area=length*width;
   return (area);
}
```

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