OBJECT ORIENTED PROGRAMING - INHERITANCE

Đào Anh Vũ

Content

- Introduction
  - Structured programming
  - Concept of OOP
  - Classes & Objects
  - Features of OOP
    - ✓Abstraction & Encapsulation
    - ✓Inheritance
      - Multiple inheritance
      - Abstract class
      - Interface
    - ✓Polymorphism

Structured Programming

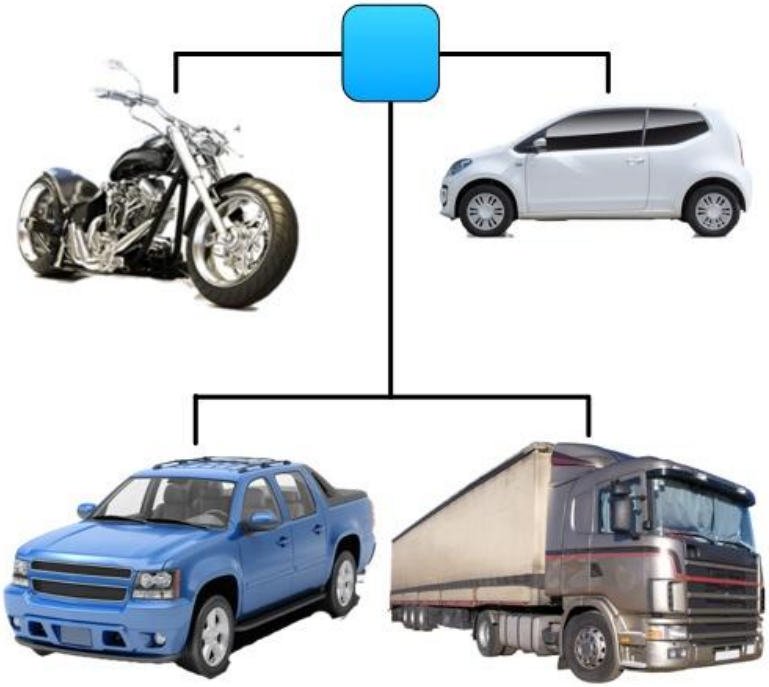
- Program = Data structure + Algorithm

Structured Programming:

- Low – level
- Focus on actions (action oriented)
- Less abstract

CONCEPTS OF OOP

- Motivation:
  - Interactions in the real world are object-object interactions.
  - We recognize everything as objects.



CLASSES & OBJECTS

- Class
  - Is blueprint from which objects are created.
  - Define data and action of objects.
- Object
  - Consist of data and actions.
  - Objects are instances of classes.
  - In most cases we interact with object through its methods.

Examples

Java

```java
public class ChessItem {
    private boolean isAlive;
    private int x;
    private int y;

    public  ChessItem() {
        isAlive = true;
        x = 0;
        y = 0;
    }
}
```

C++

```cpp
class ChessItem {
    private:
        int isAlive;
        int x;
        int y;

    public:
        ChessItem();
};
```

CLASSES & OBJECTS

- Access modifier:
  - This is the way to specify the accessibility of a class and its members with respective to other classes.
  - Access modifiers support for OOP features
  - Used at 2 levels:
    - ✓Top – level for Class & Interface.
    - ✓Member – level

CLASSES & OBJECTS

- Top – level for Class & Interface:
  - Public
  - Package/Default modifiers

Access modifier	Scope
Public	Inside and outside the package
Package/default	Just inside the package

CLASSES & OBJECTS

- Object member - Level

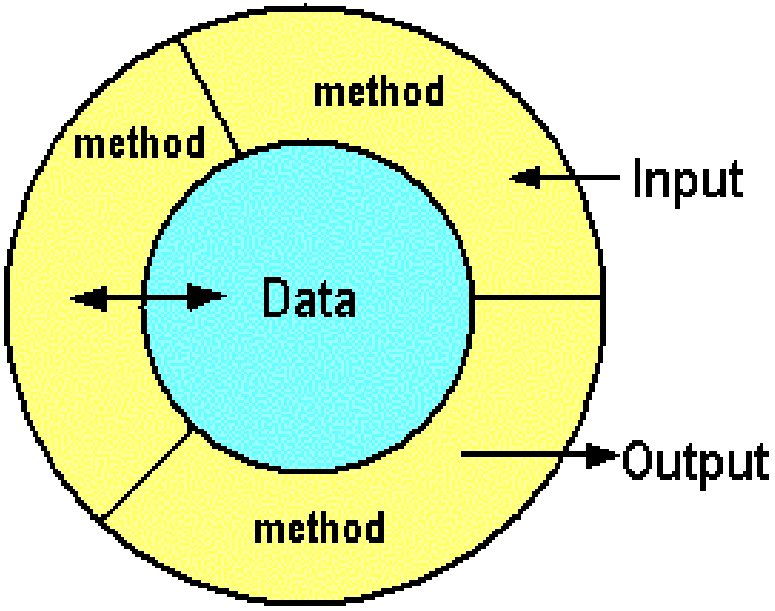
Access	public	protected	private
Same class (base)	Yes	Yes	Yes
Derived class	Yes	Yes	No
Outside classes	Yes	No	No

Abstraction

- Focus on the meaning.
  - Suppress irrelevant “implementation” details.
- See objects from outside of them.

Encapsulation

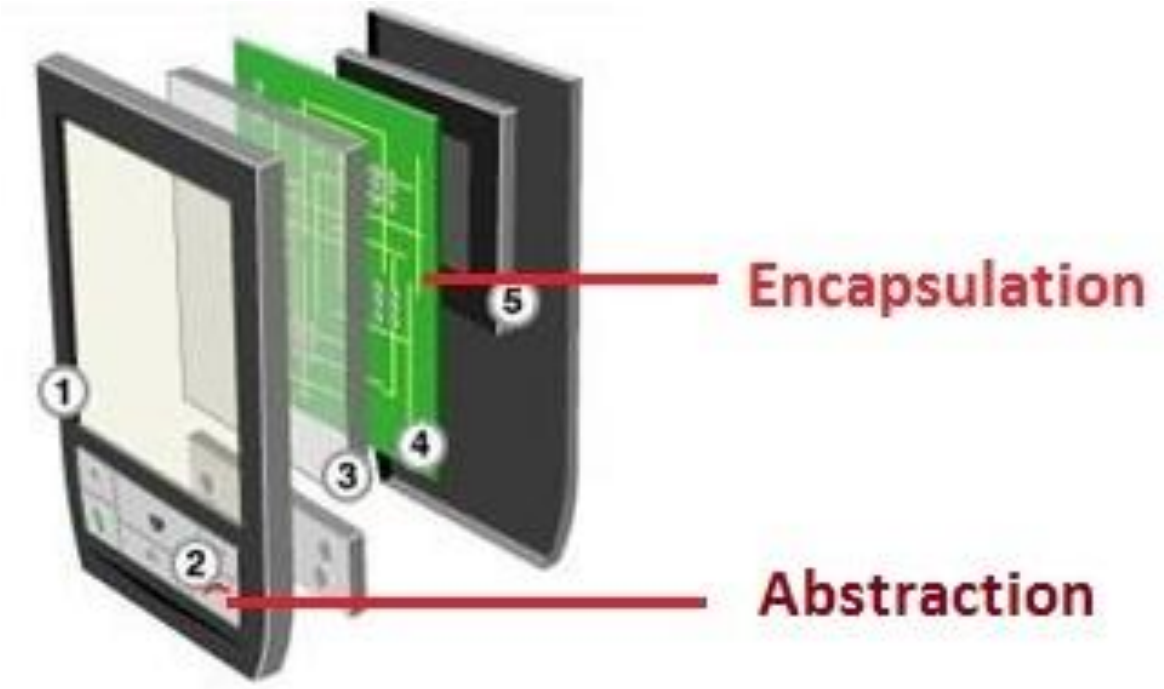
- Just methods of an object can access its own data.
- This is used to enforce the principle of data hiding.
- Handle with the visibility of object's members.
- Implement "Abstraction".



Encapsulation

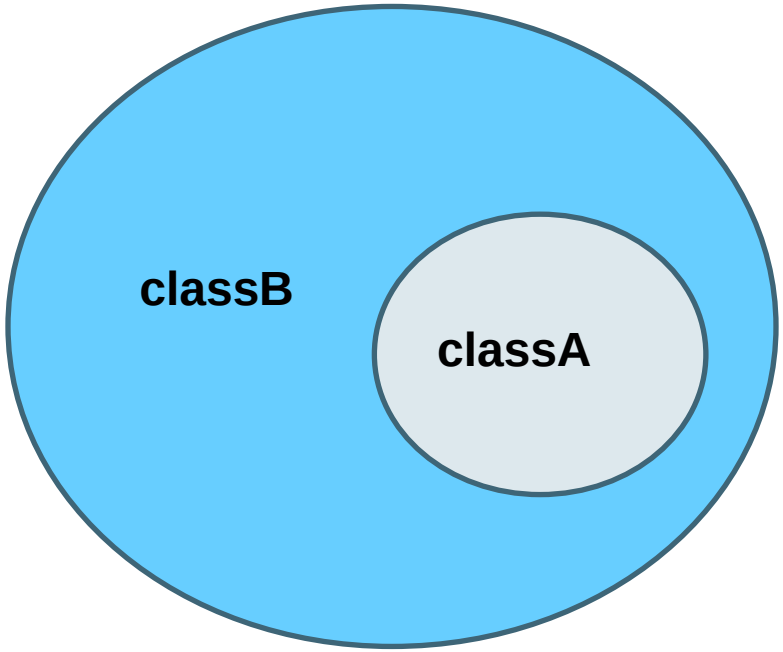
Why we make it's **harder** for our program to access its data?

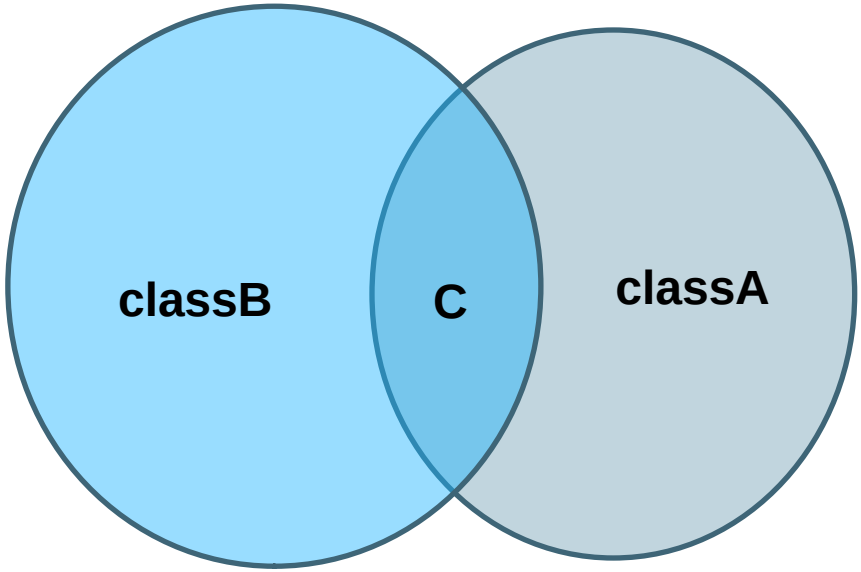
Encapsulation vs Abstraction



INHERITANCE

There are many cases that an object acquires some/all properties/methods of another object.

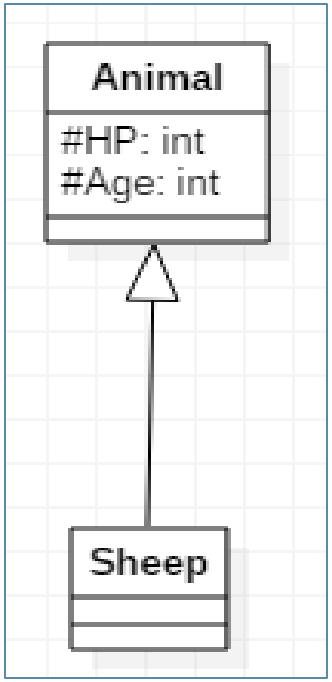




INHERITANCE

Define a new class base on existing classes.

- Existing class: base class
- New class: derived class



INHERITANCE

```java
public class Animal {
    protected int HP;
    protected int age;

    public int getHP() {
        return HP;
    }

    public void setHP(int hp) {
        this.HP = hp;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

```java
package com.myfarm.entity;

public class Sheep extends Animal {

}
```

INHERITANCE

The ***final*** keyword in Java

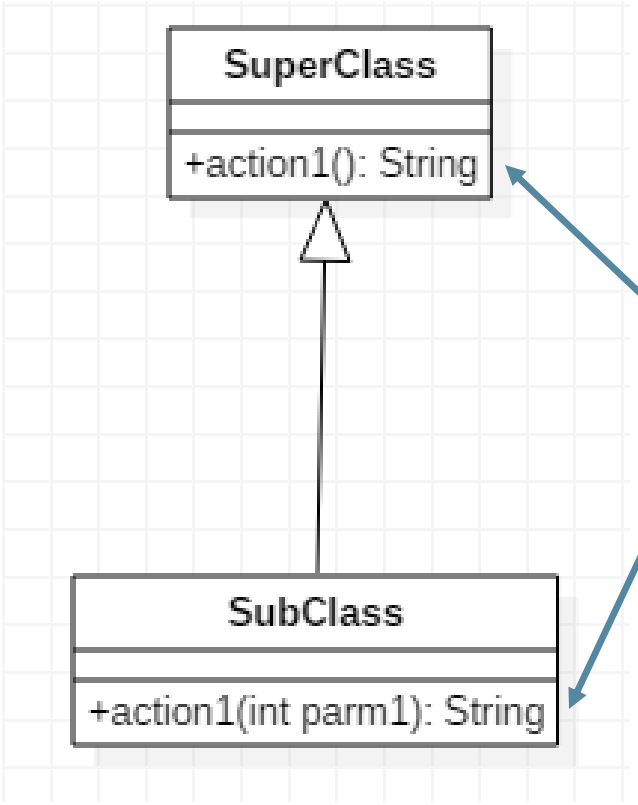
- Sometime we don't want a specific class to be a super class.
- Authors control the use of their code.

```
public final class ItemHolder {
    //
}
```

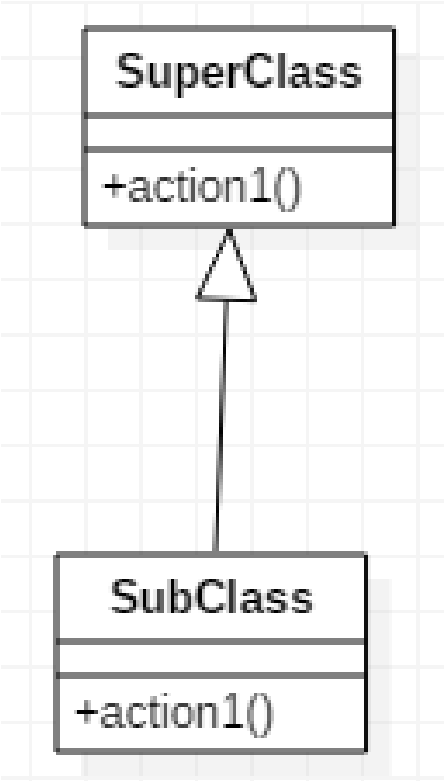
ACCESS SCOPE

Access	public	protected	private
Same class (base)	Yes	Yes	Yes
Derived class	Yes	Yes	No
Outside classes	Yes	No	No

METHOD OVERLOADING



same name but not the parameter list, type of parameters or return type.

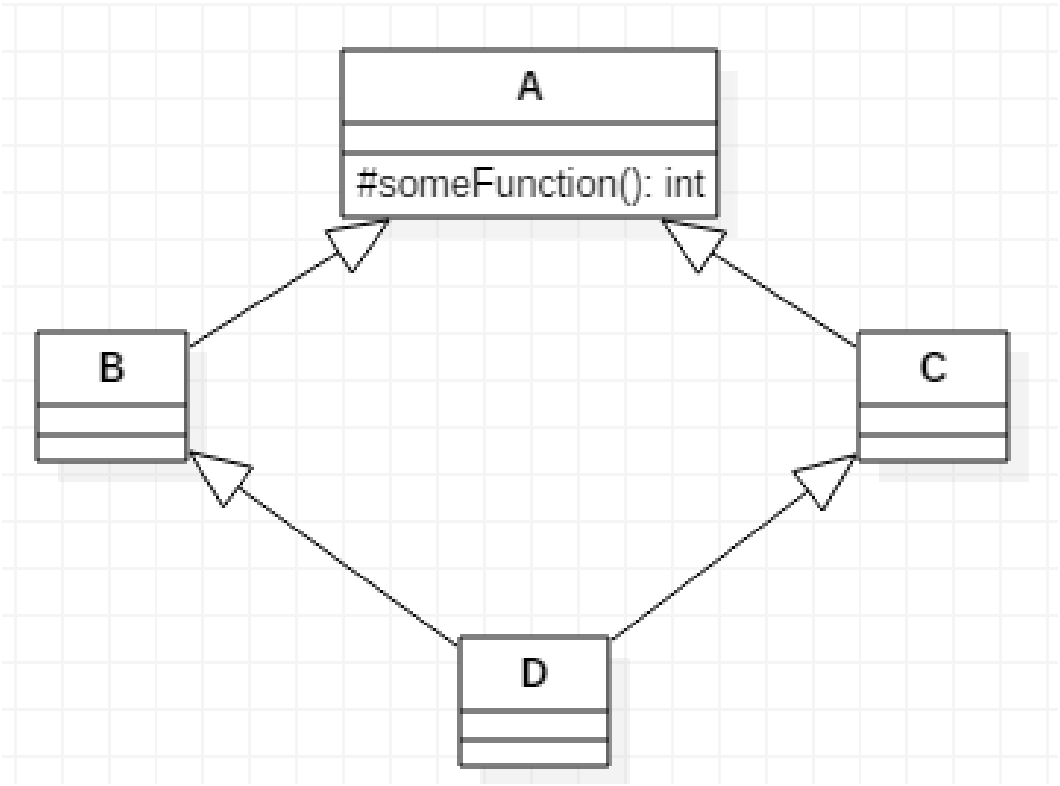


How about overriding???

MULTIPLE INHERITANCE

- A class can inherit from more than one class.
- Some languages those support multiple inheritance:
  - C++
  - Common Lisp
  - Perl
- Java does not support multiple inheritance, but we can overcome this by using *interface*.

DIAMOND PROBLEM



- Pre-Java 8 does not allow multiple inheritance → we will not face this problem.
- For C++, we solve this problem by **virtual** inheritance.

Abstract Classes

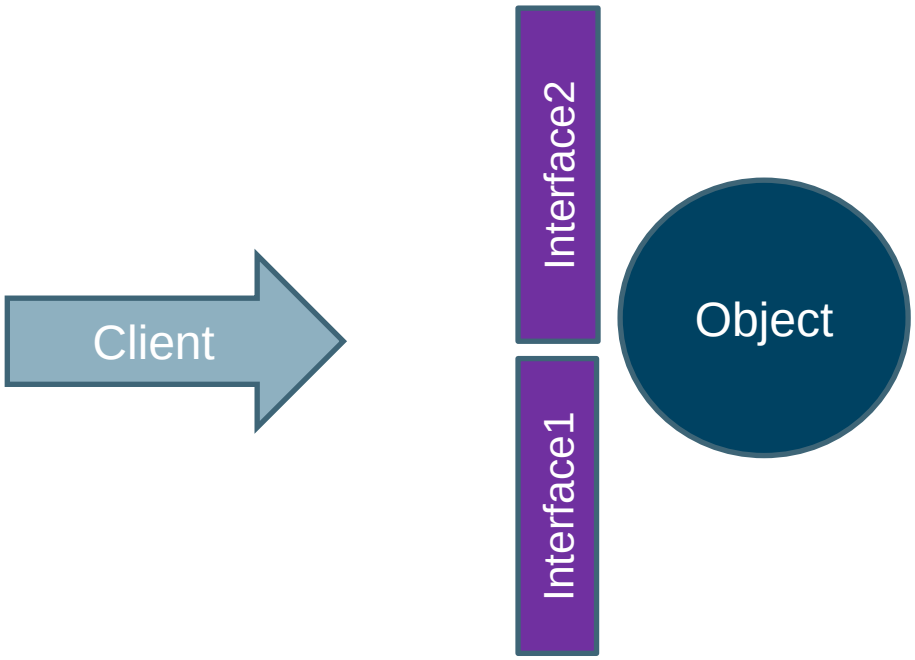
- May or may not include abstract methods.
- Cannot be instantiated

Interfaces

- A reference type in Java.
- Interfaces are not classes.
- Can contain only constant and method signatures.
- Cannot be instantiated.

Polymorphism

- Object can be represented in many forms.
- In each form, it take a specified set of action.



Polymorphism

```
public class Animal {
    protected int HP;
    protected int age;

    public int getHP() {
        return HP;
    }
}

public class Sheep extends Animal implements ICattle {
    private int fleece;

}
```

```
public int feed(Animal a) {
    //some code here
}
```

Exercises

References

- Stephen Prata, **C++ Primer Plus**, 4thEdition, *Sams*, 2004.
- Bjarne Stroustrup, **The C++ Programming Language**, 4thEdition, Addison-Wesley, 2013
- Marshall Cline, **C++ FAQ Lite**, 2ndEdition, Addison-Wesley, 2000
- Robert L. Kruse, Alexander J. RybaData, **Data Structures and Program Design in C++**, *Prentice-Hall*, Inc., 2000
- Bruce Eckel, **Thinking in C++**,*Prentice Hall*, 1998
- http://faculty.orangecoastcollege.edu/sgilbert/book/03-4-ObjectOrientedConcepts-B/index.html