OBJECT-ORIENTED LANGUAGE AND THEORY

14. WRAP-UP

Nguyen Thi Thu Trang
trangntt@soict.hust.edu.vn

1

Any question on OO Techniques?

- Abstraction, Encapsulation, Data Hiding
  - Object vs Class
  - Attribute/Field, Method
  - Method Overloading
- Object Initialization & Usage
  - Constructor
  - Operation vs Method
- Association, Aggregation, Composition
- Inheritance, Generalization
  - Method Overriding
- Interface vs Abstract Class
- Polymorphism

2

Any questions?

- UML
  - Use case diagram
  - Class diagram
- Case Study (hands-on lab)
  - Aims Project
- Mini-project
- Exam

3

Exam Structure

- Part 1
  - Short questions
  - Short exercises
- Part 2
  - Given the requirement for a program
  - Draw Use case diagram
  - Draw Class diagram
  - Write source code for a part of the program

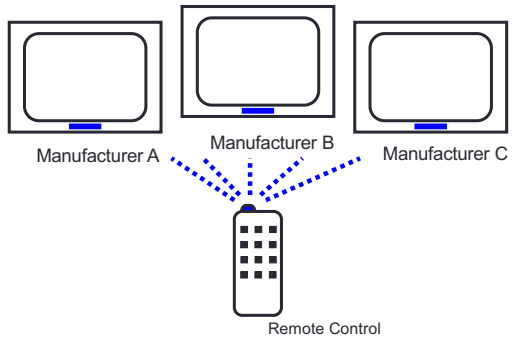
4

Comparison

- Object vs Class
- Operation vs Method
- Method vs Message
- Abstract Class vs Interface
- Aggregation vs Inheritance
- Association vs Aggregation
- Aggregation vs Composition
- Association vs Dependency
- **Call Function vs Send Message**
- Method Overloading vs Method Overriding

Why Polymorphism?

- The ability to hide many different implementations behind a single interface




```
interface TVInterface {
      public void turnOn();
      public void volumnUp(int steps); …
}
class TVA implements TVInterface {
      public void turnOn() { … }
      …
}
class TVB implements TVInterface {…}
class TVC implements TVInterface {…}
class RemoteControl {
      TVInterface tv;
      RemoteControl(TVInterface tv){setTV(tv);}
      void setTV(TVInterface tv){
            this.tv = tv;
      }
      volumnUp(int steps){ tv.volumnUp(steps); }
}
```

```
RemoteControl remote = new RemoteControl(new TVA());
remote.volumnUp(3);
```


Polymorphism

- Polymorphism: multiple ways of performance, of existance
- Polymorphism in OOP
  - Method polymorphism:
    - Methods with the same name, only difference in argument lists => method overloading
  - Object polymorphism
    - **Multiple types:** A single object to represent multiple different types (upcasting and downcasting)
    - **Multiple implementations/behaviors:** A single interface to objects of different types (upcasting+overriding – dynamic binding)

Case study in Hands-on Lab

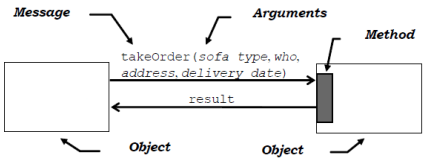
- Existing classes
  - DVD
  - Cart
  - Aims
- More classes/interfaces
  - Book
  - CD
  - Track
  - Playable
  - Disc

Function call vs. Message passing

- Call function
  - Indicate the exact piece of code to be executed.
  - Has only an execution of a function with some specific name.
  - There are no functions with the same name
- Message passing
  - **Request a service from an object and the object will decide what to do**
  - **Different objects will have different re-actions/behaviors for a message.**

Message vs. Method

- Message
  - Is sent from an object to another object and does not contain any piece of code to be executed
- Method
  - Method/function in structure programming languages
  - Is an execution of service that is requested in the message
  - Is a piece of code to be executed in order to respond to a message sent to an object




```
class A {
        private int a1;
        public float m(int i){
                ...
                return ...
        }
}

class B {
        void b(){
                A a = new A();
                a.m(9);
        }
}
```

- attribute
- behavior/operation: method signature
- **method: how?**

Objects can communicate to each other through **sending messages**