



Bài 4. Các kỹ thuật xây dựng lớp

Lý thuyết và ngôn ngữ hướng đối tượng
(bài tập)



Lab's Objectives

- In this lab, you will practice with:
 - Method overloading
 - Parameter passing
 - Classifier member vs. Instance member



Initialization of object

- When the object is generated, the member variable is assigned to the memory area, and initialized at the same time. When the developer doesn't execute the explicit initialization, an implicit initialization is done.
- number data type $\leftarrow 0$;
- reference type $\leftarrow \text{null}$
- boolean $\leftarrow \text{false}$



Constructor

- The following are the properties of a constructor:
 - Constructors have the same name as the class
 - A constructor is just like an ordinary method, however only the following information can be placed in the header of the constructor: scope or accessibility identifier (like public...), constructor's name and parameters if it has any.
 - Constructors does not have any return value
 - You cannot call a constructor directly, it can only be called by using the new operator during class instantiation.



Default constructor

- The default constructor (no-arg constructor)
 - is the constructor without any parameters.
 - If the class does not specify any constructors, then an implicit default constructor is created

- Example

```
public StudentRecord()  
{  
    //some code here  
}
```



Overload

- Define two or more method with the same name in the identical class.
- Always remember that overloaded methods have the following properties:
 - the same method name
 - different parameters or different number of parameters
 - return types can be different or the same

```
public void print( String temp ){  
    System.out.println("Name:" + name);  
    System.out.println("Address:" + address);  
    System.out.println("Age:" + age);  
}
```

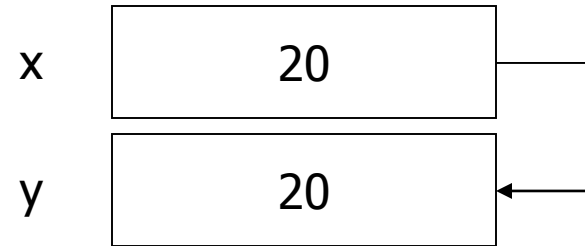
```
public void print(double eGrade, double mGrade,  
double sGrade)  
    System.out.println("Name:" + name);  
    System.out.println("Math Grade:" + mGrade);  
    System.out.println("English Grade:" + eGrade);  
    System.out.println("Science Grade:" + sGrade);  
}
```

Assignment of variable

- Basic data type

```
int x = 20;
```

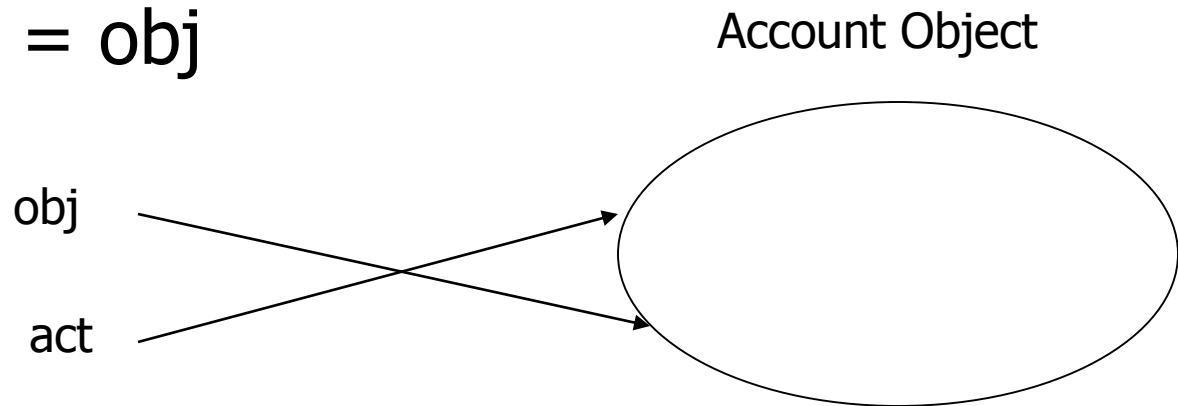
```
int y = x;
```



- Reference type

```
Account obj = new Account();
```

```
Account act = obj
```



obj and act refer the same object



The static Modifier

- We declare static methods and variables using the `static` modifier
- It associates the method or variable with the class rather than with an object of that class
- Static methods are sometimes called *class methods* and static variables are sometimes called *class variables*
- Let's carefully consider the implications of each



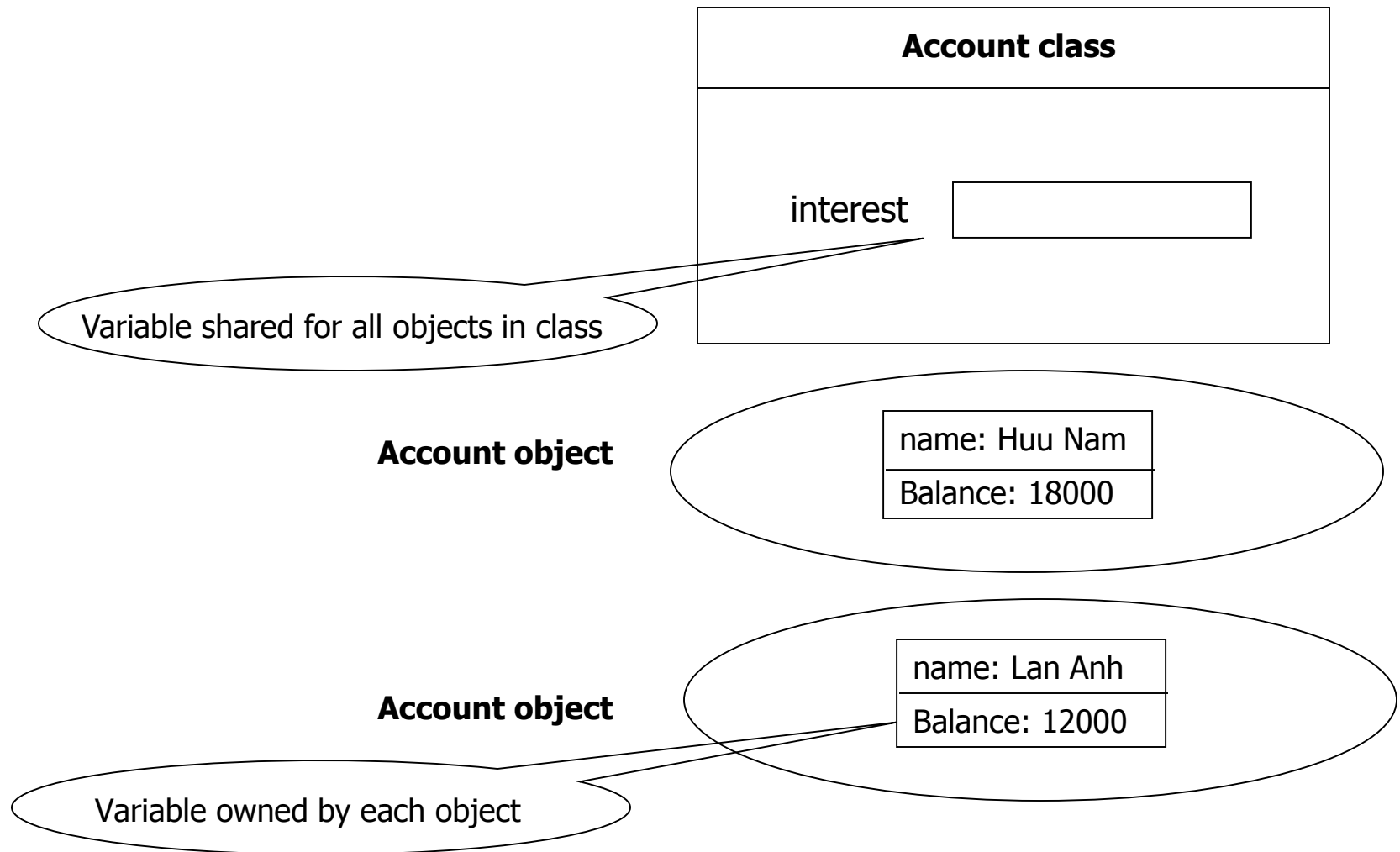
Static Variables

- Normally, each object has its own data space, but if a variable is declared as static, only one copy of the variable exists

```
private static float price;
```

- Memory space for a static variable is created when the class is first referenced
- All objects instantiated from the class share its static variables
- Changing the value of a static variable in one object changes it for all others

Static member



Static method

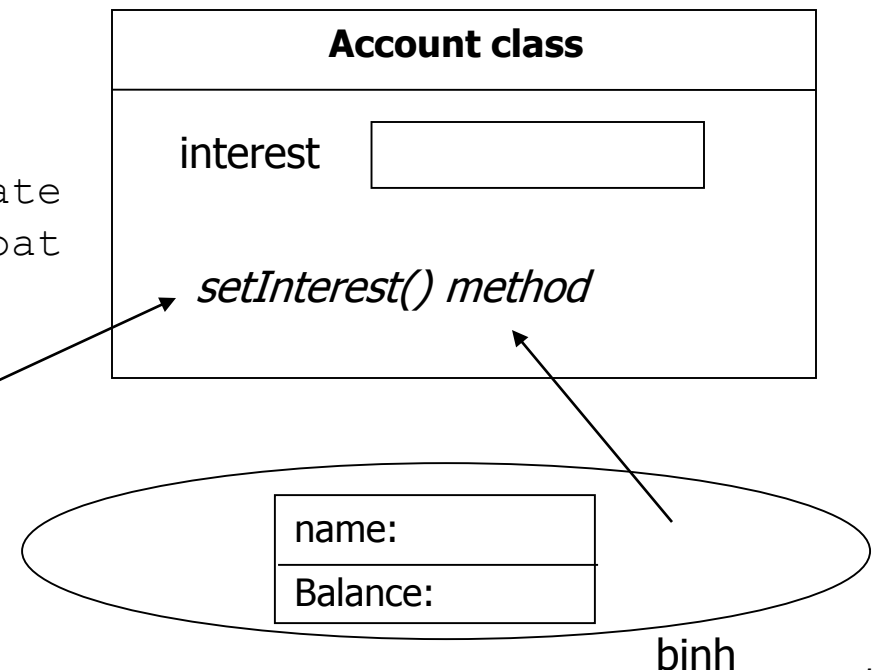
- *Declaration format:*

static Return value type Method name(Argument list){...};

- *Reference format: Class name.Method name(Argument);
or Object name.Method name(Argument);*

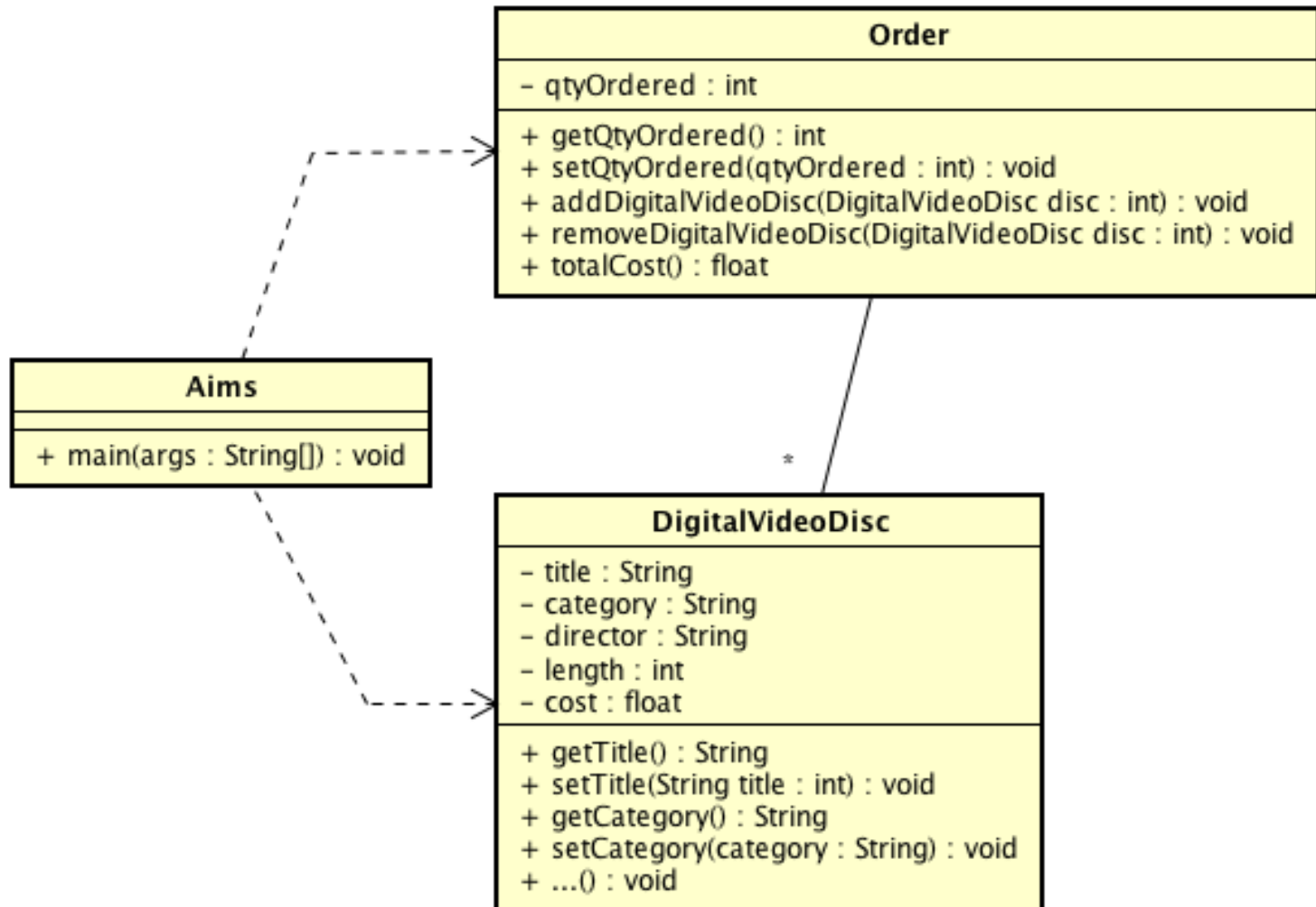
```
class Account {  
    // Member variable  
    ..  
    static float interest; // Deposit rate  
    public static void setInterest(float  
        pInterest){  
        interest = pInterest;  
    }  
}
```

```
Account.setInterest(0.05f);  
Account binh = new Account(...);  
binh.setInterest(0.04f);
```



Bài 1

- UML Class Diagram of the system





Bài 1.1

- **Open Eclipse**
- **Open the JavaProject named "AimsProject" that you have created in the previous lab.**
- **Open the class Order.java:** you will overload the method **addDigitalVideoDisc** you created last time.
 - The current method has one input parameter of class **DigitalVideoDisc**
 - You will create new method has the same name but with different type of parameter.
 - **addDigitalVideoDisc(DigitalVideoDisc [] dvdList)**
 - This method will add a list of DVDs to the current order.



Bài 1.2

- **Overloading by differing the number of parameters**
 - **Continuing focus on the Order class**
 - **Create new method named addDigitalVideoDisc**
 - The signature of this method has two parameters as following:
 - **addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)**
 - You also should verify the number of items in the current order to assure the quantity below the maximum number.
 - Inform users if the order is full and print the dvd(s) that could not be added



Bài 2. Passing parameter

- Question: ***Is JAVA a Pass by Value or a Pass by Reference programming language?***
 - **Pass by value:** The method parameter values are copied to another variable and then the copied object is passed to the method. That's why it's called pass by value
 - **Pass by reference:** An alias or reference to the actual parameter is passed to the method. That's why it's called pass by reference.

Bài 2. Passing parameter

- Practice with the **DigitalVideoDisc** class to test how JAVA passes parameter.
- Create a new class named **TestPassingParameter** in the current project

New Java Class

Java Class

⚠ The use of the default package is discouraged.

Source folder: AimsProject/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: TestPassingParameter

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Cancel Finish



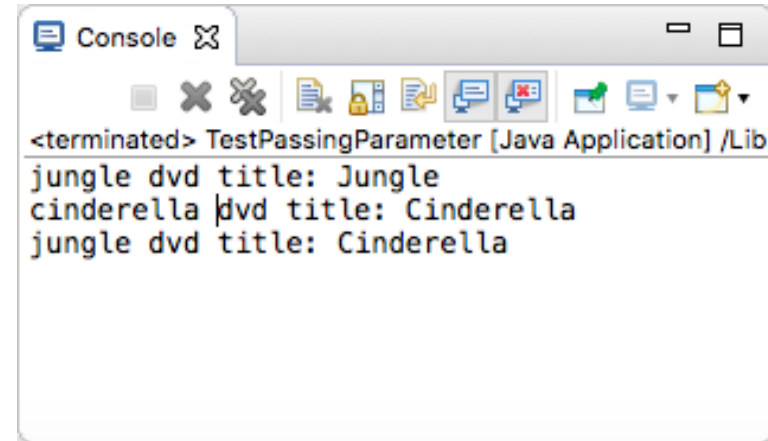
Bài 2. Passing parameter

- In the **main()** method of the class, typing the code below:

```
public class TestPassingParameter {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");  
        DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc("Cinderella");  
  
        swap(jungleDVD, cinderellaDVD);  
        System.out.println("jungle dvd title: " + jungleDVD.getTitle());  
        System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());  
  
        changeTitle(jungleDVD, cinderellaDVD.getTitle());  
        System.out.println("jungle dvd title: " + jungleDVD.getTitle());  
    }  
  
    public static void swap(Object o1, Object o2) {  
        Object tmp = o1;  
        o1 = o2;  
        o2 = tmp;  
    }  
  
    public static void changeTitle(DigitalVideoDisc dvd, String title) {  
        String oldTitle = dvd.getTitle();  
        dvd.setTitle(title);  
        dvd = new DigitalVideoDisc(oldTitle);  
    }  
}
```

Bài 2. Passing parameter

- The result in console is below:



```
<terminated> TestPassingParameter [Java Application] /Lib
jungle dvd title: Jungle
cinderella dvd title: Cinderella
jungle dvd title: Cinderella
```

- To test whether a programming language is passing by value or passing by reference, we usually use the **swap** method. This method aims to swap an object to another object.
- After the call of **swap(jungleDVD, cinderellaDVD)** why does the title of these two objects still remain?
- After the call of **changeTitle(jungleDVD, cinderellaDVD.getTitle())** why is the title of the JungleDVD changed?



Bài 3. Classifier Member and Instance Member

■ Classifier/Class member:

- Defined in a class of which a single copy exists regardless of how many instance of the class exist.
- Objective: to have variables that are **common** to all objects
- Any object of class can change the value of a class variable that's why you should always be careful with the side effect of class member
- Class variables can be manipulated without creating an instance of the class

■ Instance/Object member:

- Associated with only objects
- Defined inside the class but outside of any method
- Only initialized when the instance is created
- Their values are unique to each instance of a class
- Lives as long as the object does



Bài 3. Classifier Member and Instance Member

- **Open the Order class:**
- You should note that there are 2 instance variables
 - **itemsOrdered**
 - **qtyOrdered**
- You add a new instance variable named "**dateOrdered**" to store the date-time the ordered created.
- Add getter/setter methods for this instance variable
- This variable instance has a unique value to each instance of the **Order** class and should be initialized inside the constructor method of the **Order**.



Bài 3. Classifier Member and Instance Member

- Now we suppose that, the application only allows to make a limited number of **orders**. That means: if the current number of orders is over this limited number, users cannot make any new order.
 - Create a class attribute named "**nbOrders**" in the class **Order**
 - Create also a constant for limited number of **orders** per user for this class

```
public static final int MAX_LIMITED_ORDERS = 5;
```

```
private static int nbOrders = 0;
```

- Each time an instance of the **Order** class is created, the **nbOrders** should be updated. Therefore, you should update the value for this class variable inside the constructor method and check if **nbOrders** is below to the **MAX_LIMITED_ORDERS**.



Bài 3. Classifier Member and Instance Member

- Creating a new method to printing the list of ordered items of an order, the price of each item, the total price and the date order. Formatting the outline as below:

```
*****Order*****
```

```
Date: [date-order]
```

```
Ordered Items:
```

```
1. DVD - [Title] - [category] - [Director] - [Length]: [Price] $
```

```
2. DVD - [Title] - ...
```

```
Total cost: [total cost]
```

```
*****
```

- In the main method of the class Aims:
 - Creating different orders
 - For each order, add different items (DVDs) and print the order to the screen
 - Write some code to test what you have done in the main method