Object-Oriented Language and Theory

Lecturer: NGUYEN Thi Thu Trang, trangntt@soict.hust.edu.vn
Teaching Assistant: DO Minh Hieu, hieudominh@hotmail.com

Lab 3: Encapsulation and Class Building

* Objectives:

In this lab, you will practice with:

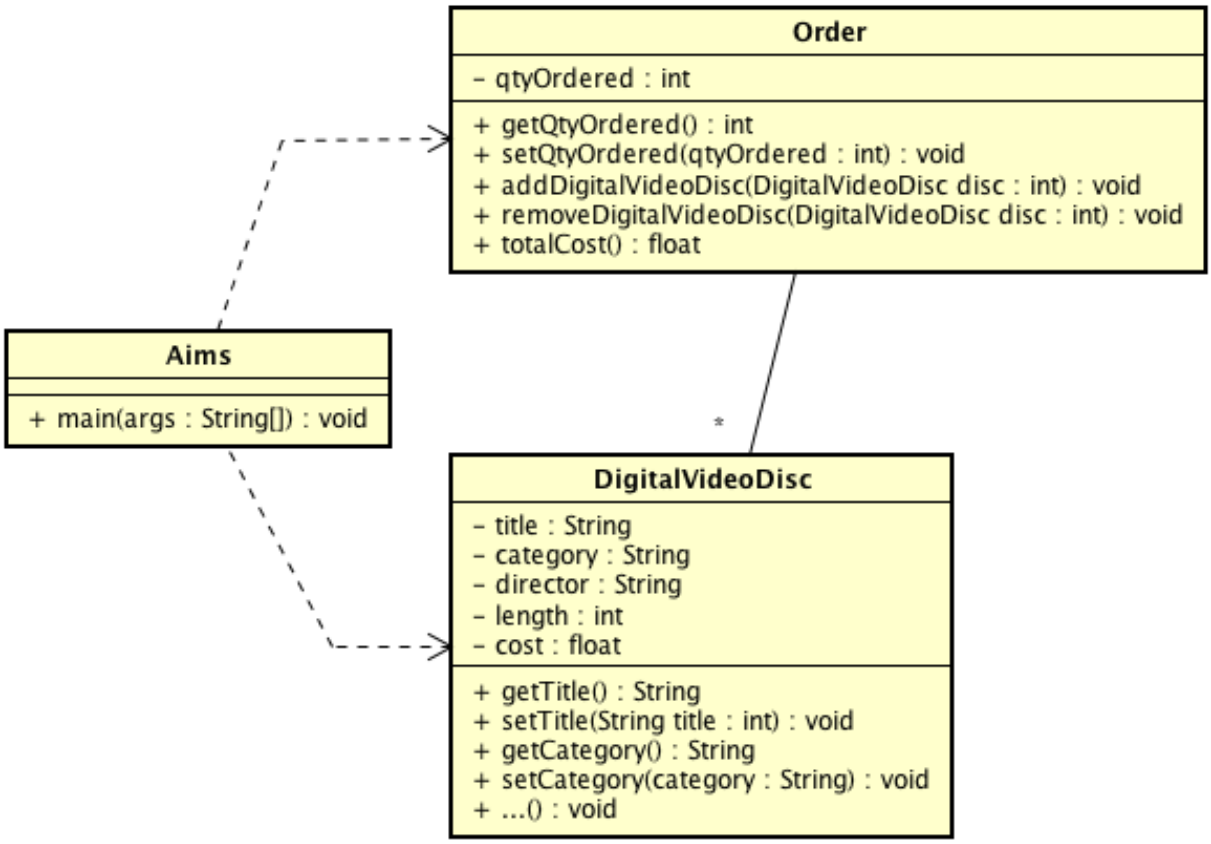
- Creating classes in Eclipse
- Constructors
- Getters and setters
- Create instances of classes.

This lab and next labs concentrate on a single project: using Eclipse to implement "AIMS: An Internet Media Store" - A system for creating orders of CDs, DVDs and books. Other exercises cover specific Object-Oriented Programming or Java topics.

* UML Class Diagram of the system

The system is console-application. There are 3 classes:

- The Aims class which provides a main() method which interacts with the rest of the system
- The DigitalVideoDisc class which stores the title, category, cost, director and length
- The Order class to maintain an array of these DigitalVideoDisc objects



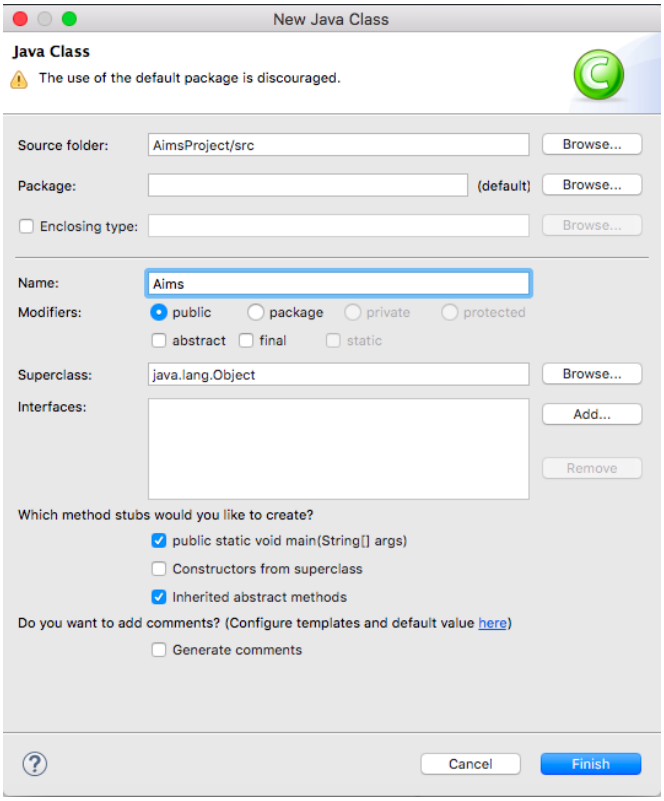
1. Create Aims class

- Open Eclipse

- Create a new JavaProject named "AimsProject"

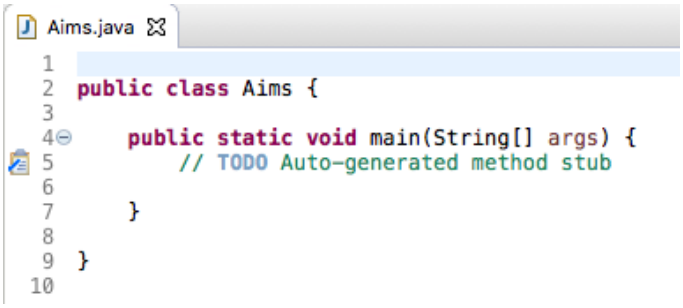
- Create Aims class: In the src folder, create a new class named Aims:

+ Right click on the folder and choose New -> Class:

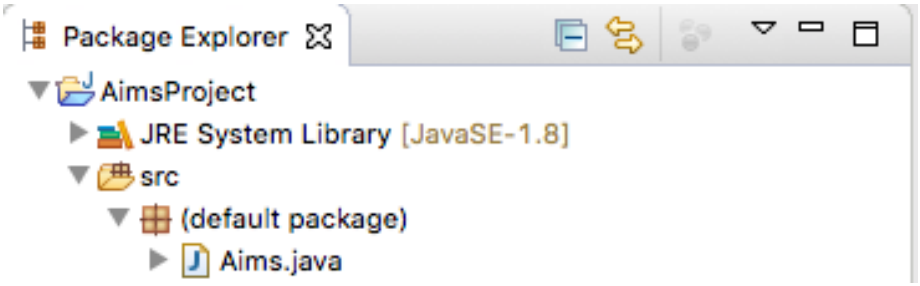


+ You may need to check the option "`public static void main(String[] args)`"

This will automatically generate the main function in the class Aims.java as the following result.



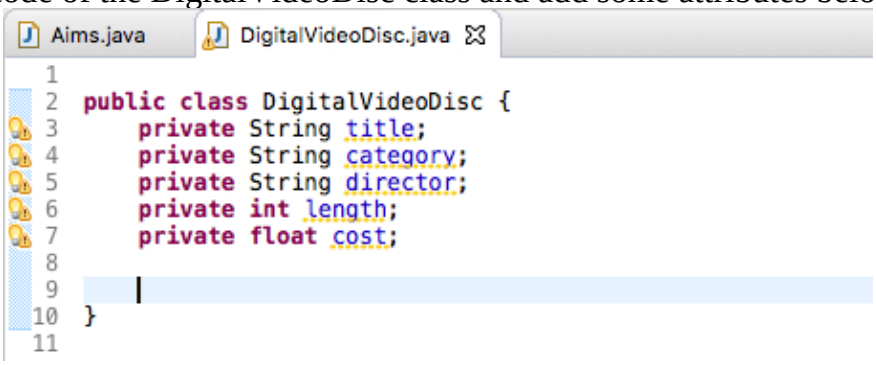
+ Because you did not choose any package for the Aims class, Eclipse then displays the icon package and mentions (default package) for your class.



+ You can create a package and move the class to this package if you want. In the folder scr, a sub-folder will be created (with the name of the package) to store the class. Do it yourself and open the src folder to see the result.

2. Create the DigitalVideoDisc class and its attributes

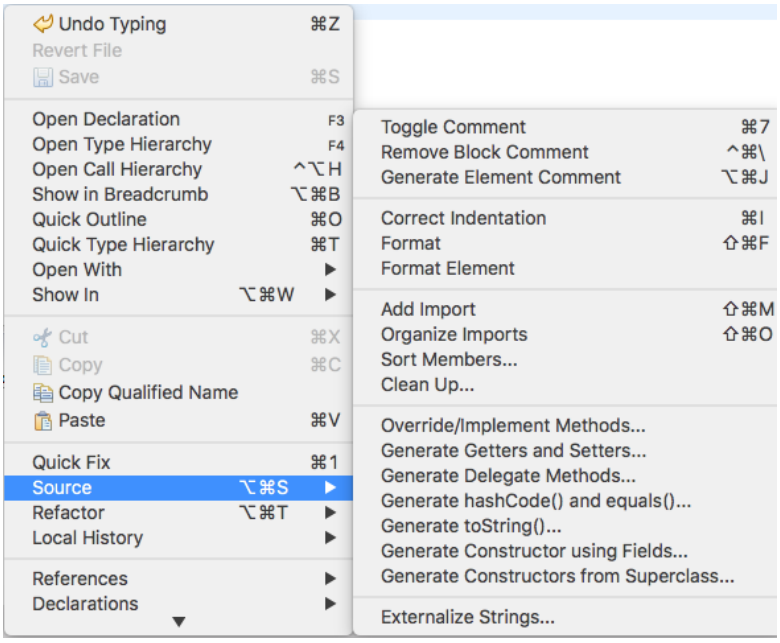
Make sure that the option for the main method is not checked.
Open the source code of the DigitalVideoDisc class and add some attributes below:

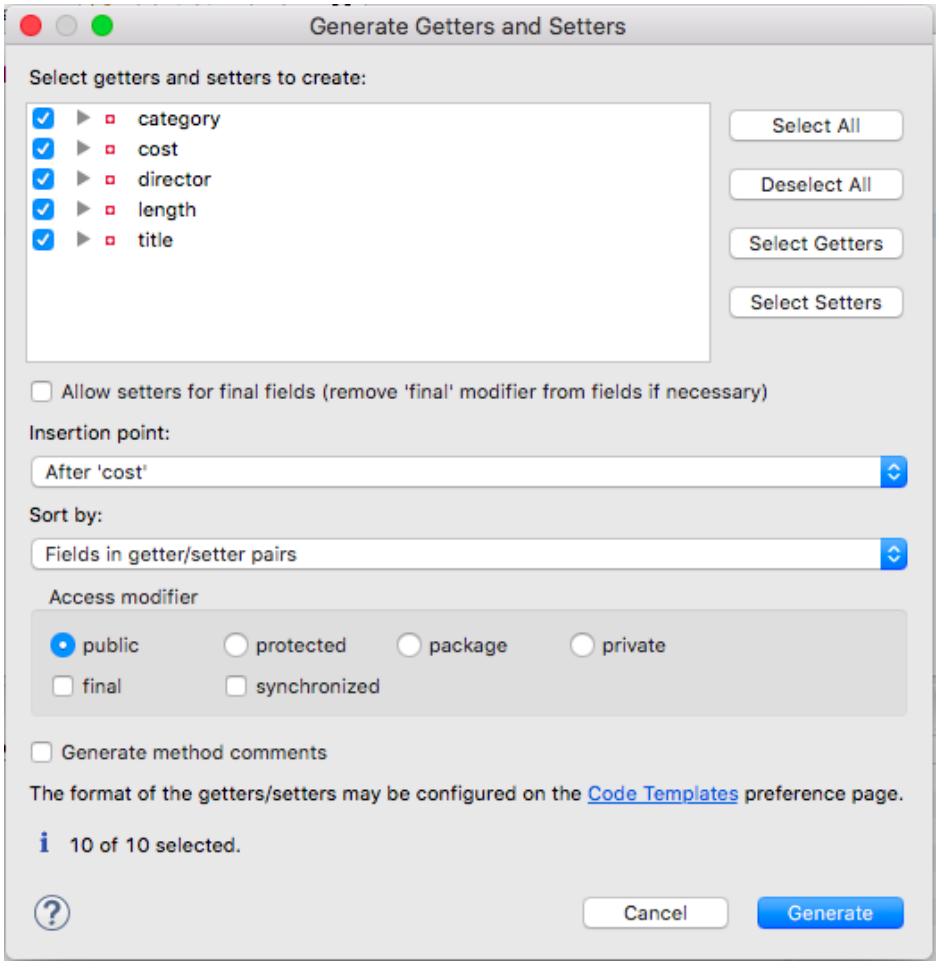


3. Create accessors and mutators for the class DigitalVideoDisc

To create setters and getters for private attributes, you can create methods to allow controlled public access to each of these private variables. Eclipse allows you to do this automatically. However, in many cases, you donot allow to create accessors and mutators for all attributes, but depending on the business. E.g. in a bank account, the balance cannot be modified directly through a mutator, but should be increase or decrease through credit or debit use cases.

- **Right click anywhere in the source file of DigitalVideoDisc.**
- **Choose Source, then choose Generate Getters and Setters**
- **Choose all the attributes**
- **Choose the option "public" in the Access modifier**
- **Click Generate**


Undo Typing ⌘Z
Revert File
Save ⌘S
Open Declaration F3
Open Type Hierarchy F4
Open Call Hierarchy ^⌥H
Show in Breadcrumb ⌥⌘B
Quick Outline ⌘O
Quick Type Hierarchy ⌘T
Open With
Show In ⌥⌘W
Cut ⌘X
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Quick Fix ⌘1
Source ⌥⌘S
Refactor ⌥⌘T
Local History
References
Declarations
Toggle Comment ⌘7
Remove Block Comment ^⌘\
Generate Element Comment ⌥⌘J
Correct Indentation ⌘I
Format ⇧⌘F
Format Element
Add Import ⇧⌘M
Organize Imports ⇧⌘O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate hashCode() and equals()...
Generate toString()...
Generate Constructor using Fields...
Generate Constructors from Superclass...
Externalize Strings...


Generate Getters and Setters
Select getters and setters to create:
category
cost
director
length
title
Select All
Deselect All
Select Getters
Select Setters
Allow setters for final fields (remove 'final' modifier from fields if necessary)
Insertion point:
After 'cost'
Sort by:
Fields in getter/setter pairs
Access modifier
public
protected
package
private
final
synchronized
Generate method comments
The format of the getters/setters may be configured on the Code Templates preference page.
10 of 10 selected.
Cancel
Generate

```java
public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public String getCategory() {
    return category;
}
public void setCategory(String category) {
    this.category = category;
}
public String getDirector() {
    return director;
}
public void setDirector(String director) {
    this.director = director;
}
public int getLength() {
    return length;
}
public void setLength(int length) {
    this.length = length;
}
public float getCost() {
    return cost;
}
public void setCost(float cost) {
    this.cost = cost;
}
```

4. Create Constructor method

By default, all classes of Java will herit from the java.lang.Object. If a class has no constructor method, this class in fact uses the constructor method of java.lang.Object. Therefore, you can always create an instance of class by a no-arguments constructor method. For example:

```
DigitalVideoDisc dvd1 = new DigitalVideoDisc();
```

In this part, you will create yourself constructor method for DigitalVideoDisc for different purposes:

- Create a DVD object by title
- Create a DVD object by category and title
- Create a DVD object by director, category and title
- Craete a DVD object by all attributes: title, category, director, length and cost

Each purpose will be corresponding to a constructor method. By doing that, you have practiced with overloading method.

Question:

- If you create a constructor method to build a DVD by title then create a constructor method to build a DVD by category. Does JAVA allow you to do this?

The first constructor method is below:

```
public DigitalVideoDisc(String title) {
    super();
    this.title = title;
}
```

You will create by yourself other constructor methods.

Eclipse also allows you to generate automatically constructor methods by field. Just do the same as generating getters and setters. Right click anywhere in the source file, Choose Source, Choose Generate constructors by fields then select fields to generate constructor methods.

5. Create the Order class to work with DigitalVideoDisc

The Order class will contain a list of DigitalVideoDisc objects and have methods capable of modifying the list.

- In this class, you will create a constant to indicate the maximum number of items that a customer can buy. Supposing that one can by maximum 10 items.
- Then, you add a field as an array to store a list of DigitalVideoDisc.

```
public class Order {

    public static final int MAX_NUMBERS_ORDERED = 10;

    private DigitalVideoDisc itemsOrdered[] = new DigitalVideoDisc[MAX_NUMBERS_ORDERED];

}
```

- To keep track of how many DigitalVideoDiscs are in the order, you must create a field named **qtyOrdered** in the Order class which stores this information.
- Create the method **addDigitalVideoDisc(DigitalVideoDisc disc)** to add more an item to the list. You should check the current number of quantity to assure that the order is not already full
- Create the method **removeDigitalVideoDisc(DigitalVideoDisc disc)** to remove the item passed by argument from the list.
- Create the **totalCost()** method which loops through the values of the array and sums the costs of the individual DigitalVideoDiscs. This method returns the total cost of the current order.

Note that, your methods should interact with users. For example: after adding it should inform to user: "The disc has been added" or "The order is almost full" if the order is full.

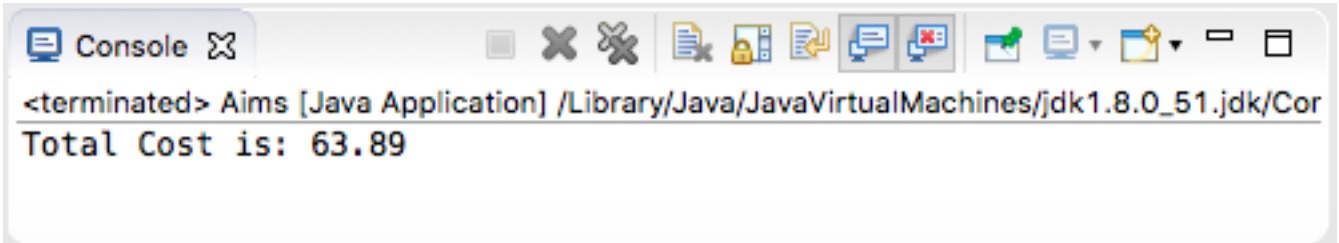
Now you have all the classes for the application. Just practice with them in the next section.

6. Create Orders of DigitalVideoDiscs:

The Aims class should create a new Order, and then create new DVDs and populate the order with those DVDs. This will be done in the main() method of the Aims class.
Do the following code in your main method and run the program to test.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Order anOrder = new Order();
    // Create a new dvd object and set the fields
    DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King");
    dvd1.setCategory ("Animation");
    dvd1.setCost (19.95f);
    dvd1.setDirector ("Roger Allers");
    dvd1.setLength (87);
    // add the dvd to the order
    anOrder.addDigitalVideoDisc(dvd1);

    DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars");
    dvd2.setCategory ("Science Fiction");
    dvd2.setCost (24.95f);
    dvd2.setDirector ("George Lucas");
    dvd2.setLength (124);
    anOrder.addDigitalVideoDisc(dvd2);

    DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladdin");
    dvd3.setCategory ("Animation");
    dvd3.setCost (18.99f);
    dvd3.setDirector ("John Musker");
    dvd3.setLength (90);

    // add the dvd to the order
    anOrder.addDigitalVideoDisc(dvd3);

    System.out.print ("Total Cost is: ");
    System.out.println (anOrder.totalCost());

}
```

The result should be:



7. You have to write code in your main method to test the remove function of the Order class and check if the code is succesfully run.

8. Create `MyDate` class which includes 3 attributes: day, month, and year (`int`).

a) Write 3 constructors
   - No parameter which set three attributes as the values of the current date
   - 3 parameters of day, month and year
   - 1 String parameter which represents for a date, e.g. “February 18th 2019”

b) Write setter and getter methods for attributes of **`MyDate`**. Consider values of a valid date.

c) Write a method named **`accept()`** which asks users to enter a date (String) from keyboard. Set corresponding values for the three attributes of **`MyDate`**.

d) Write a method named **`print()`** which print the current date to the screen.

Write the main() method in a DateTest class to test all above methods of **`MyDate`** class with different scenarios.

9. Assignment Submission

You must push the directory of the project[1] **“AimsProject”**, written by yourself, to your master branch of the valid repository before the deadline announced in the class.

Each student is expected to turn in his or her own work and not give or receive unpermitted aid. Otherwise, we would apply extreme methods for measurement to prevent cheating.

[1] See how to import/export project(s) at https://wiki.eclipse.org/Importing_and_Exporting_Projects