Object-Oriented Language and Theory

Lecturer: Nguyen Thanh Hung, [hungnt@soict.hust.edu.vn](mailto:trangntt@soict.hust.edu.vn)

Teaching Assistant: Nguyen Trung Thanh (thanh.nt176874@sis.hust.edu.vn)

Lab 3: Encapsulation and Class Building

**\* Objectives:**

In this lab, you will practice with:

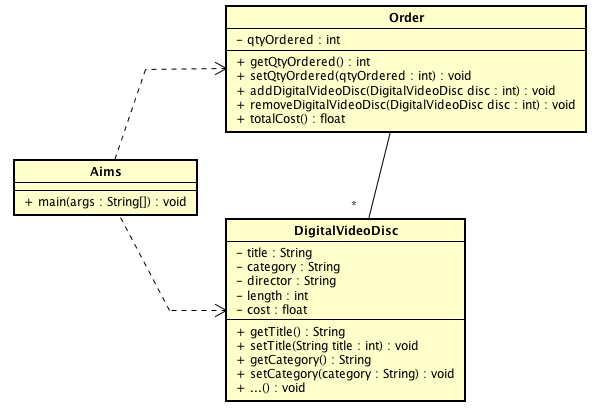
* Creating classes in Eclipse
* Constructors
* Getters and setters
* Create instances of classes.

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



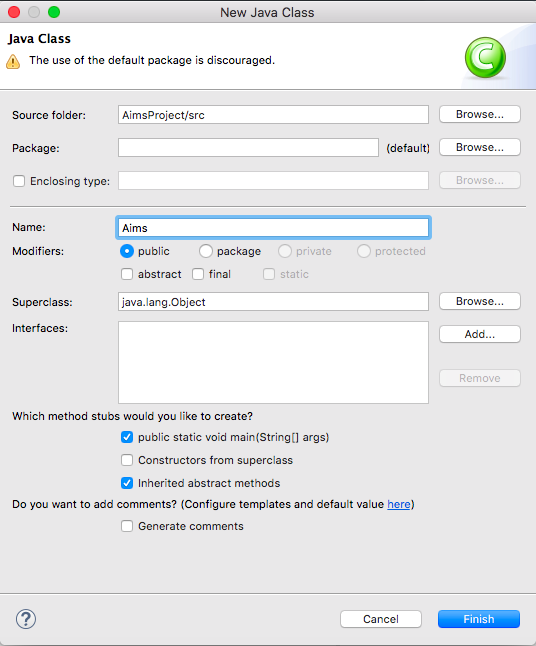
**1. Create Aims class**

**- Open Eclipse**

**- Create a new JavaProject named "AimsProject"**

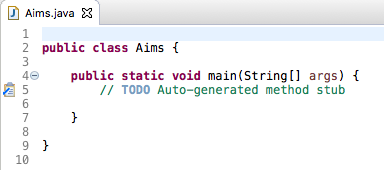
**- Create Aims class:** In the src folder, create a new class named Aims:

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

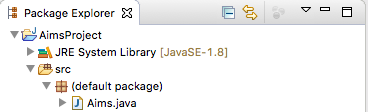
****

+ 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.



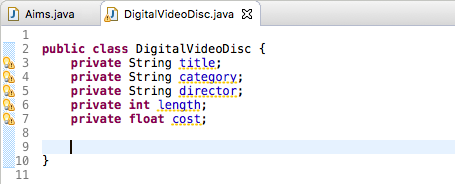
+ You can create a package and move the class to this package if you want. In the folder scr, a sub-folder will be created (with the name of the package) to store the class. Do it yourself and open the src folder to see the result.

Questions:﷽﷽﷽﷽﷽﷽﷽﷽﷽

**2. Create the DigitalVideoDisc class and its attributes (thuộc tính)**

Make sure that the option for the main method is not checked.

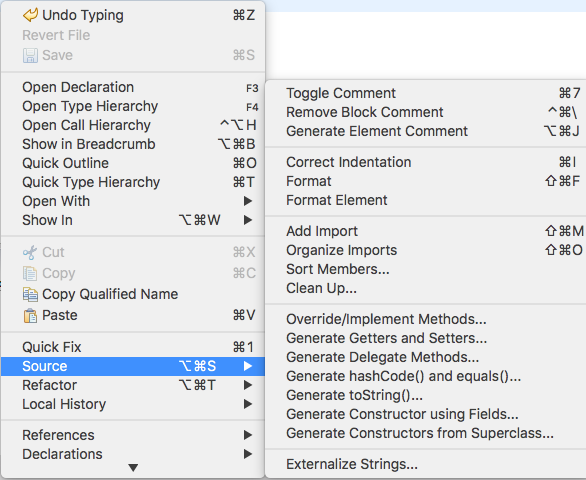
Open the source code of the DigitalVideoDisc class and add some attributes below:

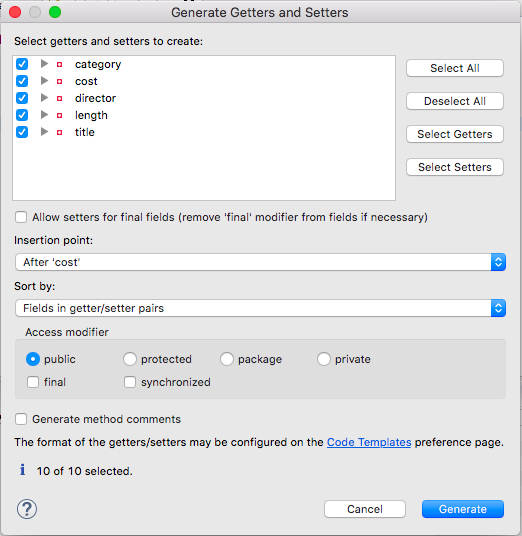


**3. Create accessors and mutators for the class DigitalVideoDisc**

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. (Intellij: Alt+insert)**
* **Choose Source, then choose Generate Getters and Setters**
* **Choose all the attributes**
* **Choose the option "public" in the Access modifier**
* **Click Generate**

****

****



**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

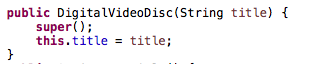
- Craete a DVD object by all attributes: title, category, director, length and cost

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:



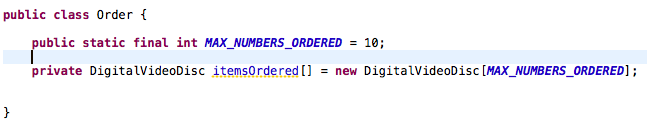
You will create by yourself other constructor methods.

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.



* 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.

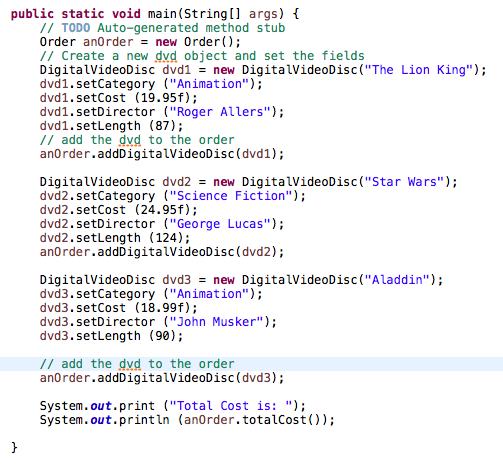
Note that, your methods should interact with users. For example: after adding it should inform to user: "The disc has been added" or "The order is almost full" if the order is full.

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

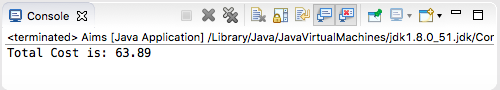
**6. Create Orders of DigitalVideoDiscs:**

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.



**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**).

* 1. 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”
  2. Write setter and getter methods for attributes of **MyDate**. Consider values of a valid date.
  3. Write a method named **accept()** which asks users to enter a date (String) from keyboard. Set corresponding values for the three attributes of **MyDate**.
  4. 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]](#footnote-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> [↑](#footnote-ref-1)