# Object-Oriented Language and Theory

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

Lab 05: Memory Management and Class Organization

## * Objectives:

In this lab, you will practice with:

- Creating packages to manage classes in Eclipse
- Using some common packages/classes of Java API, e.g. Wrapper classes, Math, System
- Practicing memory management with String and StringBuffer and other cases

You need to use the project that you did with the previous labs including both AimsProject and other projects.

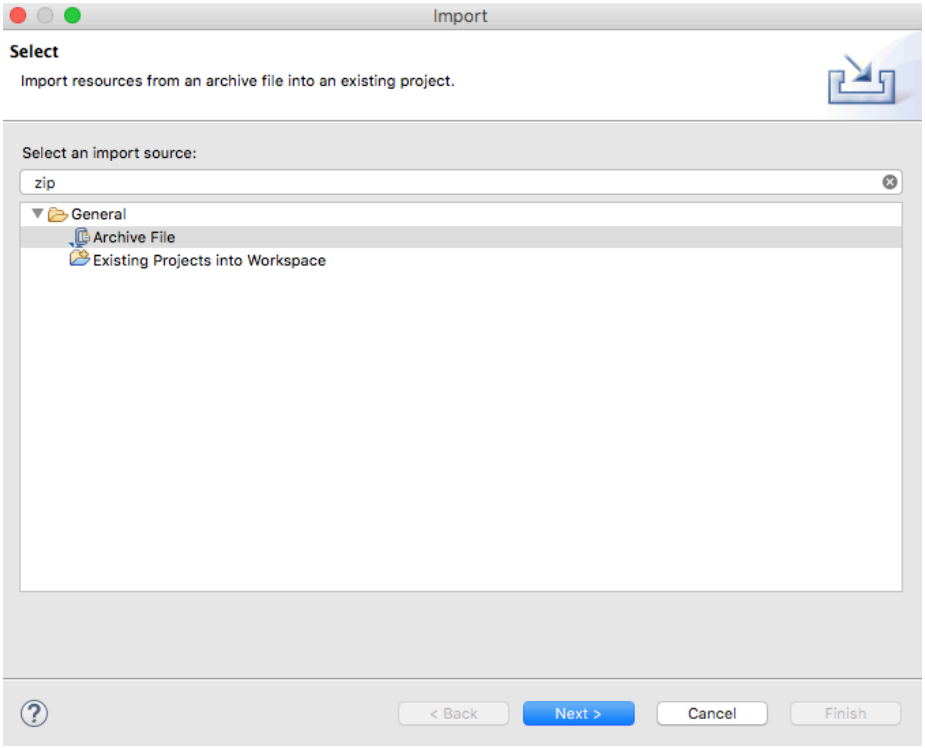
### 1. Import/export a project

**- Open Eclipse**

- You can import/export a project from/to an archive file. For example, if you want to import from a zip file, you can follow the following steps:

+ Open File -> Import.

+ Type zip to find Archive File if you have exported as a zip file before. You may choose Existing Projects into Workspace if you want to open an existing project in your computer. Ignore this step if the AimsProject is already opened in the workspace.




- Click Next and Browse to a zip file or a project to open

**2. Re-organize your projects**

- Rename project, use packages and re-organize all hands-on labs and exercises from the Lab01 up to now.

+ For renaming an item (i.e. a project, a class, a variable…), right click to the item, choose Refactor -> Rename and follow the steps to rename for all references.

+ For creating a package, right click to the project (or go to menu File) and choose New -> Package. Type the full path of package including parent packages, separated by a dot.

- Your final structure of your labs should be at least as below. You can create sub-packages for more efficently organizing your classes in both projects and all listing packages. All hands-on labs and excercises of lab01 and lab02 should be put in the corresponding package in the OtherProjects project.

```
+ AimsProject
     hust.soict.ictglobal.aims.disc.DigitalVideoDisc
     hust.soict.ictglobal.aims.order.Order
     hust.soict.ictglobal.aims.Aims
+ OtherProjects
     hust.soict.ictglobal.lab01
     hust.soict.ictglobal.lab02
     hust.soict.ictglobal.date
```

**3. Search a dvd in `AimsProject` project:**

- In the **`DigitalVideoDisc`** class, write a **`boolean search(String title)`** method which finds out (case insensitive) if the corresponding disk of the current object contains the **`title`**. Remember that if the **`title`** has multiple tokens (e.g. "Harry Potter"), the method still returns **`true`** if the disc has a **`title`** including all the tokens (e.g. both two tokens "Harry" and "Potter") regardless their order and their distance (so **`true`** for all title including a token "Harry" and a token "Potter" in any position in the **`title`**)

- In the **`Order`** class, write a method **`DigitalVideoDisc getALuckyItem()`** which randomly pick out (remember to use **`Math.random()`**) an item for free. Rember to update and test the methods for listing of dvds and total cost of an order (specifying a lucky and free item).

- Create **`DiskTest`** class to test these methods

4. **`String`**, **`StringBuilder`** and **`StringBuffer`**:

- In the **`OtherProjects`** project, create a new package **`hust.soict.ictglobal.garbage`**

- Create a new class **`ConcatenationInLoops`** to test the processing time to construct **`String`** using **`+`** operator, **`StringBuffer`** and **`StringBuilder`**. The following piece of code is a suggestion:

```java
public class ConcatenationInLoops {
    public static void main(String[] args) {
        Random r = new Random(123);
        long start = System.currentTimeMillis();
        String s = "";
        for (int i = 0; i < 65536; i++)
            s += r.nextInt(2);
        System.out.println(System.currentTimeMillis() - start);  // This prints roughly 4500.

        r = new Random(123);
        start = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 65536; i++)
            sb.append(r.nextInt(2));
        s = sb.toString();
        System.out.println(System.currentTimeMillis() - start);  // This prints 5.
    }
}
```

More information on **`String`** concatenation, please refer https://redfin.engineering/java-string-concatenation-which-way-is-best-8f590a7d22a8.

- Create a new class **`GarbageCreator`** in the package **`hust.soict.ictglobal.garbage`**. Create "garbage" as much as possible and observe when you run a program (it should let the program stop working when too much "garbage". Write another class **`NoGarbage`** to solve the problem.

Some suggestions:

- Read a text file to a **`String`** without using **`StringBuffer`** to concatenate String (only use **`+`** operator). Observe and capture your screen when you choose a very long file
- Improve the code using **`StringBuffer`**