Object-Oriented Language and Theory

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

Lab 07: Abstract Class and Interface

* Objectives:

In this lab, you will practice with:

- Creating abstract class, abstract methods
- Creating interface and implement it

You need to use the project that you did with the previous labs including both AimsProject and other projects. Please follow the Release Flow to commit and push to the bitbucket.

1. Import the AimsProject

- Open Eclipse

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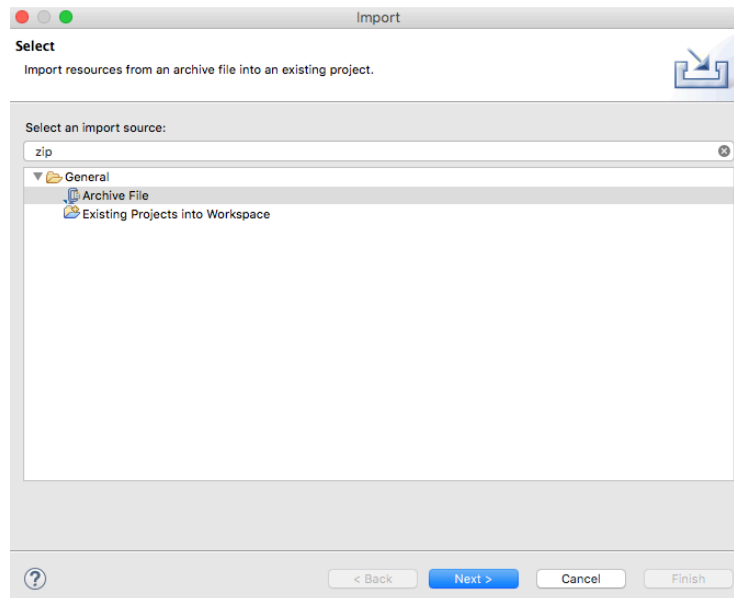
- Click Next and Browse to a zip file or a project to open

2. Extending the AIMS project to allow the ordering of CDs (Compact Discs)

As with **DigitalVideoDisc** and **Book**, the **CompactDisc** class will extend **Media**, inheriting the **title**, **category** and **cost** fields and the associated methods.

You can apply Release Flow here by creating a **topic** branch for making the **Media** class abstract.

2.1. Make the `Media` as an abstract class

- Open **Media** class
- Modify the **Media** class and make it **abstract**
- Keep three fields of **Media**: **title**, **category**, **cost** and their associated methods, but remove the setters.

2.2. Create the `Disc` class extending the `Media` class

- The **Disc** class has two fields: **length** and **director**
- Create **getter** methods for these fields
- Create constructor(s) for this class. Use super() if possible.
- Make the **DigitalVideoDisc** extending the **Disc** class. Make changes if need be.
- Create the **CompactDisc** extending the **Disc** class. Save your changes.

2.3. Create the `Track` class which models a track on a compact disc and will store information incuding the `title` and `length` of the track

- Add two fields: **String title** and **int length**
- Make these fields **private** and create their **getter** methods as **public**
- Create constructor(s) for this class.
- Save your changes

2.4. Open the `CompactDisc` class

- Add 2 fields to this class:
  - a **String** as **artist**
  - an **ArrayList** of **Track** as **tracks**
- Make all these fields as **private.** Create public **getter** method for only **artist**.
- Create constructor(s) for this class. Use super() if possible.
- Create methods **addTrack**() and **removeTrack**()
  - The **addTrack**() method should check if the input track is already in the list of tracks and inform users
  - The **removeTrack**() method should check if the input track existed in the list of tracks and inform users
- Create the **getLength**() method
  - Because each track in the CD has a length, the length of the CD should be the sum of lengths of all its tracks.
- Save your changes

3. Create the `Playable` interface

The **Playable** interface is created to allow classes to indicate that they implement a **play**() method. You can apply Release Flow here by creating a **topic** branch for implementing **Playable** interface.

- Create **Playable** interface, and add to it the method prototype: **public void play();**
- Save your changes
- Implement the **Playable** with **CompactDisc**, **DigitalVideoDisc** and **Track**
  - For each of these classes **CompactDisc** and **DigitalVideoDisc**, edit the class description to include the keywords **implements Playable**, after the keyword **extends Disc**
  - For the **Track** class, insert the keywords **implements Playable** after the keywords **public class Track**
- Implement **play**() for **DigitalVideoDisc** and **Track**
  - Add the method **play**() to these two classes
  - In the **DigitalVideoDisc**, simply print to screen:

```
public void play() {
    System.out.println("Playing DVD: " + this.getTitle());
    System.out.println("DVD length: " + this.getLength());
}
```

  - Similar additions with the **Track** class
- Implement **play**() for **CompactDisc**
  - Since the **CompactDisc** class contains a **ArrayList** of **Tracks**, each of which can be played on its own. The **play**() method should output some information about the **CompactDisc** to console
  - Loop through each track of the arraylist and call **Track's** play() method

4. Update the Aims class
- You will update the **Aims** class to test your changes.
  - Create more choices for your application
  - Update the menu of choices:
    - For the addition of new item to the order, the program should ask for the type: **Book**, **CompactDisc** or **DigitalVideoDisc**
    - For the **CompactDisc**, the program should allow to add information of **Tracks**
    - When adding a cd/dvd to the order, the user may ask for play them
- You will update the class diagram for the above classes and interfaces.

5. Update the menu of the main method in Aims
Open the Aims class. You will create a prompted menu as following:
```
public static void showAdminMenu() {
```

```
        System.out.println("Product Management Application: ");
        System.out.println("-------------------------------");
        System.out.println("1. Create new item");
        System.out.println("2. Delete item by id");
        System.out.println("3. Display the items list");
        System.out.println("0. Exit");
        System.out.println("-------------------------------");
        System.out.println("Please choose a number: 0-1-2-3");
}
```

Now, you have to create a menu for admin who can create item (DVD, CD, Track, Book) and add them to a list of available items for the store. You have to note that, because of the difference between DVD, CD and Book, the program should allow to create each item in a different way.