Object-Oriented Language and Theory

Bui Thi Mai Anh, anhbtm@soict.hust.edu.vn

Lab 10: Exception Handling

* Objectives:

In this lab, you will practice with:

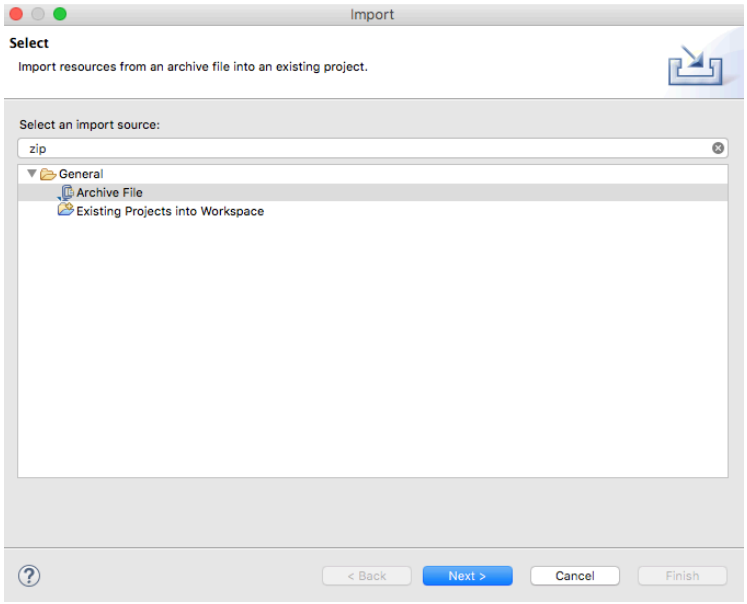
- Create various Exception types
- Raise exceptions
- Catch and report exceptions

In this lab, you will create a subclass of **`Exception`** called **`PlayerException`**. This exception is raised when one of the **`Media`** subclasses' **`play`**() method encounters a **`length`** of 0. The **`play`**() method will be altered to use **`try-catch`** syntax to catch the error.

1. Open the workspace and the AIMS project

- 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. Create a class which inherits from Exception

The **`PlayerException`** class represents an exception that will be thrown when an exceptional condition occurs during the playing of a media in your **`AimsProject`**.

2.1. Create new class named `PlayerException`

- Enter the following specifications in the New Java Class dialog:
  - Name: **`PlayerException`**
  - Package: **`hust.soict.ictglobal.aims`**
  - Access Modifier: **`public`**
  - Superclass: **`java.lang.Exception`**
  - Constructor from Superclass: checked
  - **`public static void main(String [] args)`:** do not check
  - All other boxes: do not check
- Finish

2.2. Raise the `PlayerException` in the `play()` method

- Update **`play()`** method in **`DigitalVideoDisc`** and **`Track`**
  - For each of **`DigitalVideoDisc`** and **`Track`**, update the **`play()`** method to first check the object's length using **`getLength()`** method. If the length of the **`Media`** is less than or equal to zero, the **`Media`** object cannot be played.
  - At this point, you should output an error message using **`System.err.println()`** method and the **`PlayerException`** should be raised.
- For example, the code for the **`play`**() of **`DigitalVideoDisc`** should be:

```
public void play() throws PlayerException {
  if (this.getLength() <= 0) {
    System.err.println("ERROR: DVD length is 0");
    throw (new PlayerException());
  }
  System.out.println("Playing DVD: " + this.getTitle());
  System.out.println("DVD length: " + this.getLength());
}
```

- Save your changes and make the same with the `play()` method of `Track`.

2.3. Update `play()` in the `Playable` interface

- Change the method signature for the **`Playable`** interface's **`play()`** method to include the throws **`PlayerException`** keywords.

2.4. Update `play()` in `CompactDisc`

- The **`play`**() method in the **`CompactDisc`** is more interesting because not only it is possible for the **`CompactDisc`** to have an invalid **`length`** of 0 or less, but it is also possible that as it iterates through the tracks to play each one, there may have a track of length 0 or less
- First update the **`play`**() method in **`CompactDisc`** class to check the length using **`getLength`**() method as you did with **`DigitalVideoDisc`**

- Output an error message using **`System.err.println()`** method and then raise the **`PlayerException`**. Be sure to change the method signature to include throws **`PlayerException`** keywords.
- Update the **`play()`** method to catch a **`PlayerException`** raised by each **`Track`** using block **`try-catch.`**

The code should be as following:

```
public void play() throws PlayerException {
  if (this.getLength() <= 0) {
    System.err.println("ERROR: CD length is 0");
    throw (new PlayerException());
  }

  System.out.println("Playing CD: " + this.getTitle());
  System.out.println("CD length:" + this.getLength());

  java.util.Iterator iter = tracks.iterator();
  Track nextTrack;

  while (iter.hasNext()) {
    nextTrack = (Track) iter.next();
    try {
      nextTrack.play();
    } catch (PlayerException e) {
      e.printStackTrace();
    }
  }
}
```

3. Update the `Aims` class

- The **`Aims`** class must be updated to handle any exceptions generated when the **`play`**() methods are called.
- Try to use **`try-catch`** block when you call the **`play`**() method of **`Media's`** objects.
- For example, each time you play a DVD or a CD, should use the **`try-catch`** block as following:

```
// Invoke the DVD's play() method
try {
  dvd1.play();
} catch (Exception e) {
  e.printStackTrace();
}
```

```
// Invoke the CD's play() method
try {
  cd1.play();
} catch (Exception e) {
  e.printStackTrace();
}
```

With all these steps, you have practiced with User-defined Exception (**`PlayerException`**), **`try-catch`** block and also **`throw`**. The **`try-catch`** block is used in the main function of class **`Aims.java`** and in the **`play()`** function of the **`CompactDisc.java`**.

4. Practical Exercise

Create a program to read from command line students' information including: studentID, studentName, birthday(format: dd/mm/yyyy), gpa (float number from 0 to 4). Remember that you should have a class Student with constructors and getters/setters.

You have to create your own exception class:

- **`IllegalBirthDayFormatException`** to check if the format of input birthday is wrong. The illegal day or month should also cause this exception happening.
- **`IllegalGPAException`** to check if the input gpa is not between 0.0 to 10.0.