Lab 10 Exception Handling

Lý thuyết và ngôn ngữ hướng đối tượng (bài tập)

Lab’s Objectives

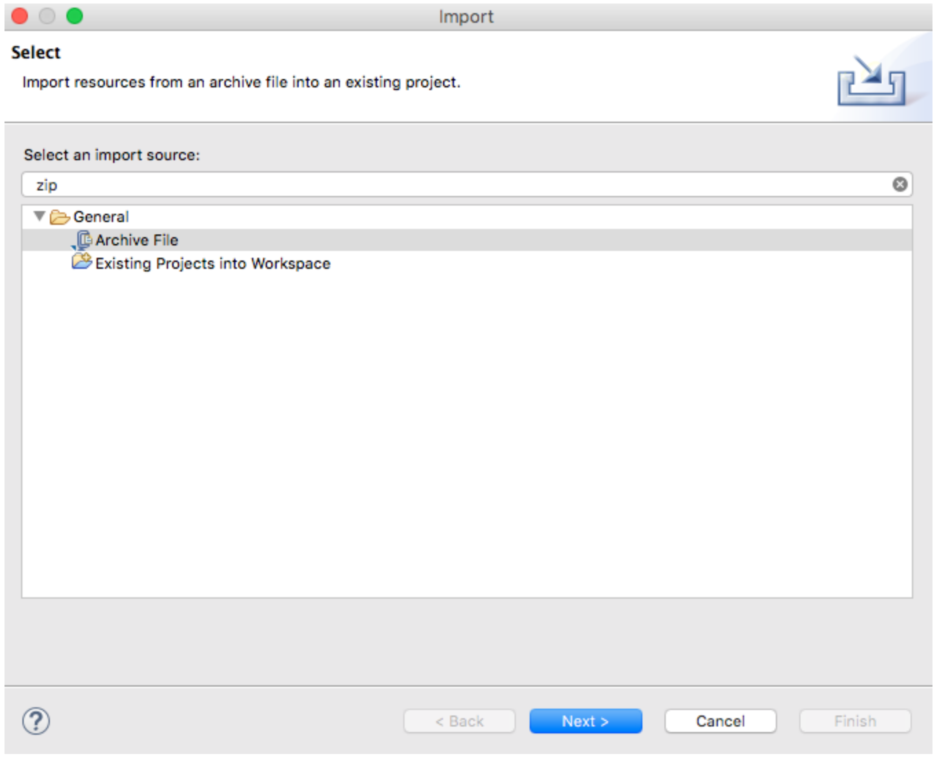
- In this lab, you will practice with:
  - Create various Exception types
  - Raise exceptions
  - Catch and report exceptions

Lab’s Objectives

- 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.


Import

Select

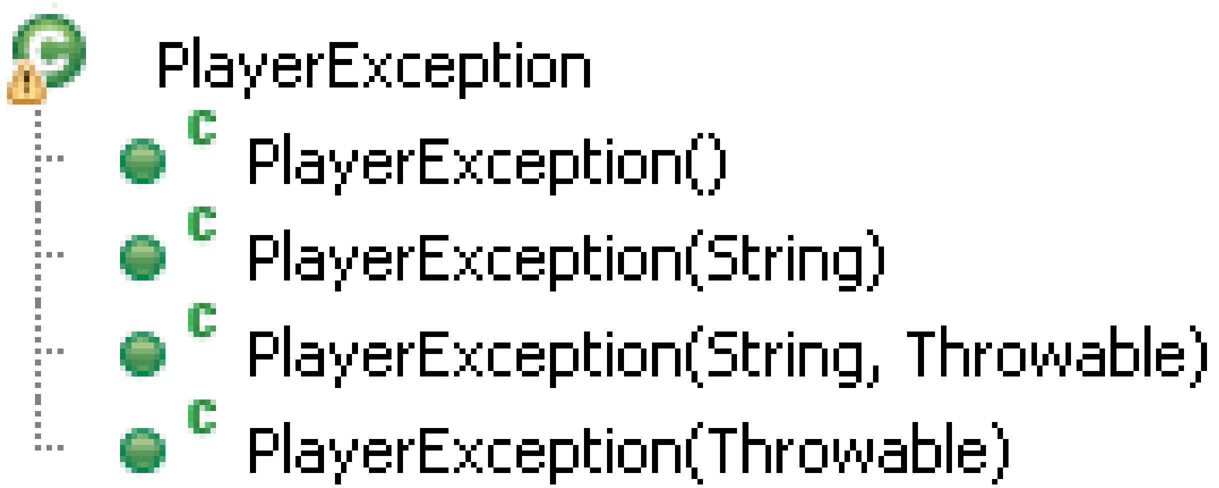
Import resources from an archive file into an existing project.

Select an import source:

zip

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**.



```
PlayerException
  PlayerException()
  PlayerException(String)
  PlayerException(String, Throwable)
  PlayerException(Throwable)
```

2. Create a class which inherits from Exception

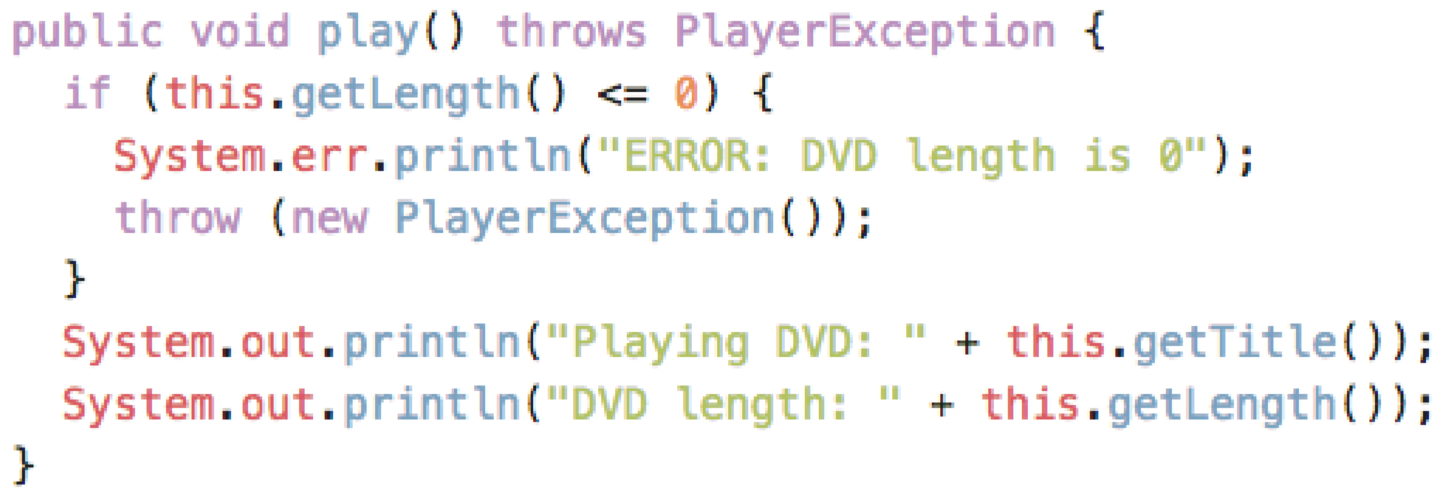
- 2.1. Create new class named PlayerException
- Enter the following specifications in the New Java Class dialog:
  - Name: **PlayerException**
  - Package: **hust.soict.ictglobal.aims**
  - Access Modifier: **public**
  - Superclass: **java.lang.Exception**
  - Constructor from Superclass: checked
 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

2. Create a class which inherits from Exception

- 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.

2. Create a class which inherits from Exception

- 2.2. Raise the PlayerException in the play() method
- 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. Create a class which inherits from Exception

- 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:

```
public interface Playable {

    public void play() throws PlayerException;

}
```

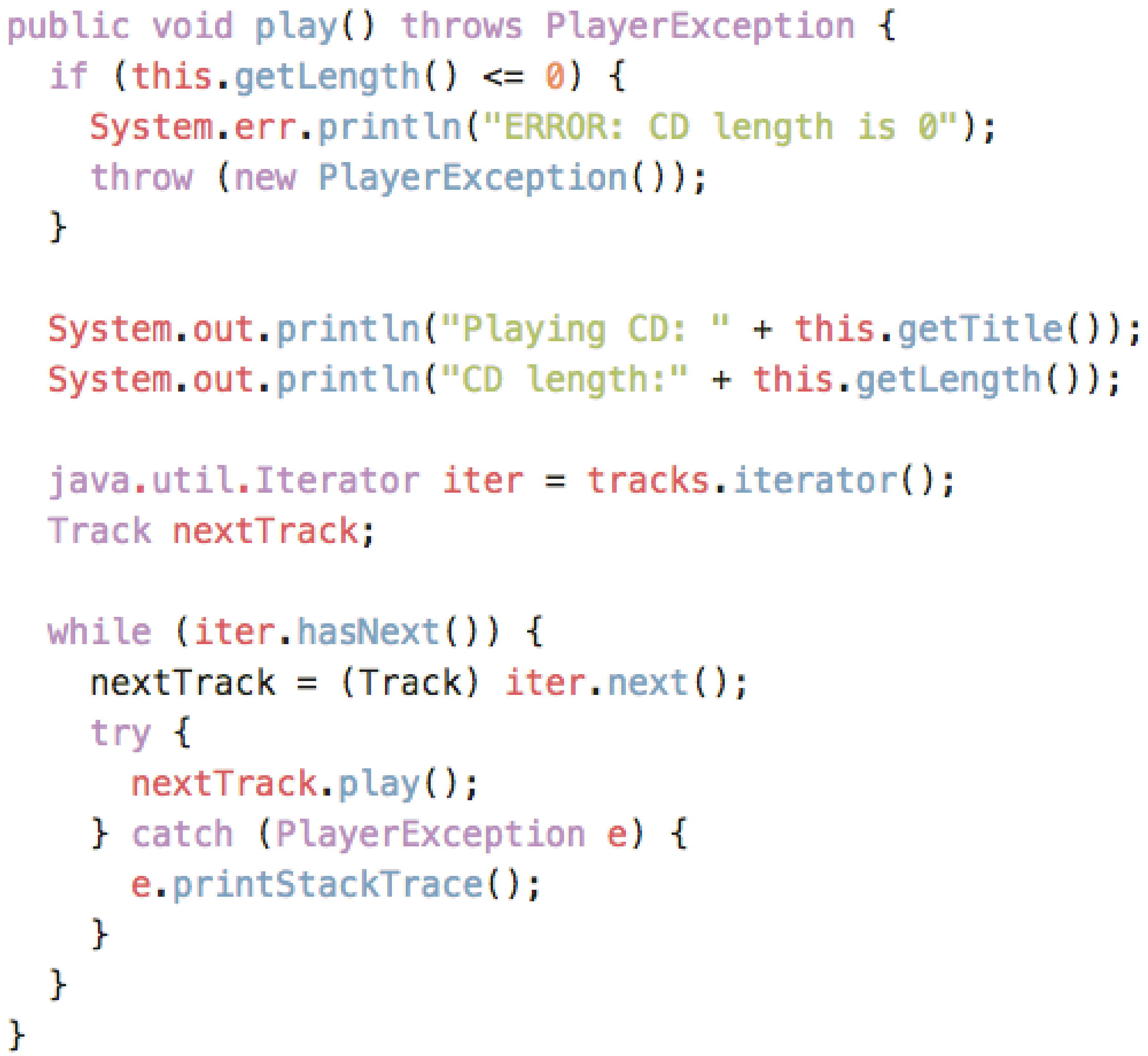
- Save your changes

2. Create a class which inherits from Exception

- 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

2. Create a class which inherits from Exception

- 2.4. Update play() in CompactDisc
  - Update the **play()** method to catch a **PlayerException** raised by each **Track** using block **try-catch**
  - You should modify the above source code so that if any track in a CD can't play, it throws an PlayerException exception



```
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. What happens when you don't update for them to catch?
- Try to use **try-catch** block when you call the **play()** method of **Media's** objects.
- 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 method of class **Aims.java** and in the **play()** method of the **CompactDisc.java**.
  - Print all information of the exception object, e.g. **getMessage()**, **toString()**, **printStackTrace()**.

4. Override the equals() method of the Object class

- Override the equals() method of the Object class and compareTo() method of Comparable for Media class
  - Two medias are equals if they have the same **title** and **cost**
  - Please remember to check for **NullPointerException** and **ClassCastException** if applicable.
  - You may use **instanceof** operator to check if an object is an instance of a **ClassType**.

5. Check all the previous source codes in previous labs

- **Check all the previous source codes in previous labs to catch/handle runtime exceptions**

6. 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:
  - **IllegalBirthDayException** 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.