CS 250 - Fall 2021 - Lab 04: Interface and File IO

Available: Sep 15, 2021.
Due date: **Sep 20, 2021, 11:59 PM**

Objectives:

The objectives of this lab are:

- To use the Comparable/Comparator interface for comparison and sorting
- To use Java I/O API for reading/writing text files
- To use Java I/O API for reading/writing binary files

Prerequisite:

- Read section 8.3 and chapter 11 of the free textbook *Java, Java, Java, 3E* and the slides of lectures 02 and 04
- Read online tutorials of Interface in Java
- Read online tutorials of Java I/O Tutorial

Part 1: Comparable/Comparator Interface for Object Comparision and Sorting

Step 1. Run the Eclipse IDE program

NOTE: Remember how to find this program because you're going to need it every week.

Step 2. Create the Lab04 project.

- Click on the "Create a java project" link in the Package Explorer window or right-click on that window then select NEW then Java project. Enter the name of your project, i.e. "Lab04". The "Location" text box should indicate that your project will be stored on the desktop. In the project JRE setting, make sure at least one Java SE version is selected, default to JavaSE-1.8 if you followed instructions on the Prerequisites section. Click on Finish to create the project.

- In the Package Explorer window, right-click on the "src" item, then select "New" and then Class. Next, create a class named “P4”.

- Copy the following code to override the generated code by Eclipse.

```
import java.util.Arrays;

public class Student {
	private String id;		//student id
	private String name;	//name of the student
	private int birthYear;	//birth year of the student

	public Student(String id, String name, int year) {
		this.id = id;
		this.name = name;
		this.birthYear = year;
	}

	public String getName() { return name; }
	public String getId() {  return id;   }
	public void setId(String id) { this.id = id;	}
	public int getBirthYear() {    return birthYear;  }
	public void setBirthYear(int birthYear) { this.birthYear = birthYear;
	}
	public void setName(String name) {    this.name = name;  }

	public String toString() {
		return "Student: " + id + ", " + name + ", " + birthYear;
	}

	public static void main(String[] args) {
		Student[] arrStudents = new Student[5];
		arrStudents[0] = new Student("157886", "Callum", 1985);
		arrStudents[1] = new Student("865427", "Blake",  1946);
		arrStudents[2] = new Student("143433", "Kayden", 1999);
		arrStudents[3] = new Student("774343", "Kason",  2001);
		arrStudents[4] = new Student("454767", "Colin",  1975);
		Arrays.sort(arrStudents);
		for(Student obj : arrStudents)
			System.out.println(obj);
	}
}
```

If you try to run the above code, there is an error (exception) on the line of Arrays.sort() where we are trying to sort the array of students. This method helps sort an array of elements. However, it supports an array of primitive data such as int, float, double, String, etc. as Java knows how to compare the values of these primitive data types. To make this method work on a custom class (Student) in this case, this class should implement the interface `Comparable`, which implements the abstract method `compareTo(Object object)`. The `Comparable` interface, which belongs to `java.lang` package, can be used to order the objects of a custom class. You can read the Java API documentation of this interface here: java.lang.Comparable interface. This interface

contains only one method named `compareTo(Object)`. When a class implements this interface, it provides the default method to determine the order of two objects that belong to this class.

Now, modify the code of this Student class to implement this interface.

- Modify the class header to add `implements Comparable` after `public class Student`. Now, Eclipse will highlight a red error on the Person class asking you to implements the required abstract method `public int compareTo(Object object)`

- Now, you need to implement the required `public int compareTo(Object object)` method. You should implement this method with the following steps:

  - The parameter of this method is a general instance of the class `Object`. The first step is casting this `object` to an instance of the `Person` class:

    ```
    Student obj = (Student)object;
    ```

  - The `Student` class defines three attributes: id, name, and birth year. We need to pickup a default method to compare and order the instance of this class, let say the `id` attribute. So, you need to compare the id(by using `getId()` method) between `this` instance and the casted `obj` instance:

    ```
    if(this.getId() == obj.getId()) return 0; //equals as they have same id

    else if(this.getId() < obj.getId()) return -1; //this < obj as it has smaller id

    else return 1;  //this > obj as it has bigger id
    ```

  - Now, verify that there are no compile errors and run the program.

Part 2: Java I/O for reading/writing files

In this part, we will practice using Java IO libraries to read the list of Student objects from a file and write the list of Students to file. The following activities guide you to complete these tasks. Create a new class named P4_IO and complete the following steps:

2.1 Create an input students.txt file with the following steps:

- Right-click on the `src` item on your Eclipse project, choose File, New
- Enter `students.txt` in the `File Name:` text box, then click Finish

- Input the following student records in the new file `students.txt` and then save it.

```
237860451, Callum, 1985
867584562, Blake,  1946
628457851, Kayden, 1999
485651247, Kason,  2001
784583166, Colin,  1975
789451263, James,  1965
245879612, Derek,  1974
544622157, James,  1999
```

2.2 Implement method to read students.txt file:

Add the following new empty method to the P4_IO class

```
/**
  * This method reads an array of Student objects from the input text file
  * @param filePath: The input file path
  * @return An array of Student object
  */
public static Student[] readStudentsFromTextFile(String filePath) {
    //TODO: add code below
}
```

The above method receives a string parameter of the file path that stores the Student records and returns an array of students. Implement this method with the following steps:

- Create a File instance as follows: `File inFile = new File( path );`
- Create the instance of `BufferedReader` wrapped with `try ... catch` as follows:

```
try (BufferedReader br = new BufferedReader(new FileReader(inFile));) {

} catch(FileNotFoundException fnfe) {

} catch(IOException ex) {

}
```

- Remember to import the required classes and packages.
- Create the String variable named `line` to store each line read from the input file: `String line;`
- As the first line in the file stores the number of Student objects, we read this line and parse it into an integer variable len
  line = br.readLine();
  int len = Integer.parseInt(line);

- Now, declare an array of students variable: Student[] arrStudents = new Student[len];
- Declare an integer count variable with an initial value 0 to count how many students read from the input file: int count = 0;
- Use a while loop inside the try block to read each line of the input stream `br`:

```
while( (line = br.readLine()) != null ) {

}
```

- The line variable stores the Student record including id, name, and birth year which are separated by a comma. We can use the `split(String regex)` to split the line into multiple tokens. As between two data fields are separated by a comma and one or more spaces. You can split the line into an array of string tokens as `String[] arr = line.split(",\\s+");`
- Create a String variable to store the id: `String id = arr[0];`
- Create a String variable to store the name: `String name = arr[1];`
- Create an int variable to store the birth year after the parsing of the second token to integer: `int bYear = Integer.parseInt(arr[2]);`
- Now, we can create an instance of the Student class: `Student obj = new Student(id, name, bYear);`
- Put this student object to the array of students at the index count: `arrPerson[count] = obj;`
- Increase the count variable: count++;
- Print an error message that the file does not exist inside the catch block of FileNotFoundException
- Print an error message that the file can not be read inside the catch block of IOException
- As the method Integer.parseInt() can throw the error NumberFormatException, we need to catch this exception and show an error message. You can add this catch at the end of this method.
- Finally, add a return statement: `return arrPerson;`

2.3 Test the method of reading students file in the main() method

Add the public static void main(String[] args) method to the P4_IO class if this method is not exist. Implement the following steps:

- Call the method `readStudentsFromTextFile` with the input file `students.txt` and get the array of Student objects from such a file: `Student[] arrStudents = readStudentsFromTextFile ( "src/students.txt" );`

- Use a for loop to display the array of Student objects. In this part we will not use the syntax `for(int i = 0; i < arr.length; i++) { ... }`. We will use the format `for(Student obj : arrStudents) { ... }`
- Inside the for loop, you can use `System.out.println` on the `object` variable

2.4 Implement a method to write an array of students to a text file:

Add the following new empty method to the P4_IO class:

```
/**
 * This method writes an array of Student objects to an output text file
 * @param path: the output file path
 * @param arrStudents: the array list of the students
 */
private static void saveStudentsToTextFile(String path, Student[] arrStudents) {
    //TODO: add code below
}
```

The above method receives a string parameter of the file path and an array of students to output them to the given text file. Implement this method with the following steps:

- Create a File instance as follows: `File outFile = new File( path );`
- Create the instance of `BufferedWriter` wrapped with `try ... catch` as follows:

```
try (BufferedWriter bw = new BufferedWriter(new FileWriter(new File(path)))) {

} catch(IOException ex) {

}
```

- Remember to import the required classes and packages.
- Use a for loop to iterate each Person object:

```
for(Students obj : arrStudents) {
}
```

- Firstly, we need to write the student's id and the comma and space separator: `bw.write(obj.getId() + ", ");`
- Secondly, we need to write the student's name and the comma and space separator: `bw.write(obj.getName() + ", ");`
- Secondly, we write the person's birth year and the comma and space separator: `bw.write(obj.getBirthYear() + ", ");`
- Finally, write the new line character: `bw.newLine();`
- Inside the catch block, print the error message such as "Can not open the file to write"

2.3 Test the method of writing students file in the main() method

At the end of the main() method to the P4_IO class, implement the following steps:

- Modify the properties of different student objects in arrStudents:

```
arrStudents[0].setName("Ryan Schmidt");
arrStudents[2].setId("18178117919");
```

- Now, all the method saveStudentsToTextFile as follows:
  `saveStudentsToTextFile("src/students_new.txt", arrStudents);`
- Run the program now, verify that the file `students_new.txt` is created successfully in the src folder of the current project. Open this file and verify its content is corrected.

Lab Assignment

Complete the following tasks:

a. Add a new field major of type String in the Student class. This field stores the current major of each student (for example Computer Science, Accounting, etc.). You need to modify the constructor method to add the initialization of this new data field. Also, add a new accessor method to return the major and modify the toString() method to print additionally the major attribute.

g. In the students.txt file, add a new column named Major and add sample values for this column

h. Modify the method readStudentsFromTextFile method to read the major attribute from the file.

i. Modify the method saveStudentsToTextFile method to export the major attribute of each Student record to file.

Lab Result Submission:

You need to submit the results of the following sections to the D2L assignment item of Lab#4:

- Complete the required Lab Assignment above

- Compress the whole Eclipse source code of your Lab#4 project in zip format using the Export function