Guided Lab - 303.13.1 - Reading a Delimited File

Introduction:

- For separating a delimited file, we can use:
    - **String class** - has a **split()** method to identify the comma delimiter and split the row into fields.
    - **Scanner class** - has a **useDelimiter()** method to identify the comma delimiter and split the row into fields.

Objective:

In this Lab, we will demonstrate how to read a Delimited file by using Java. Below is one of the processes:

- Create an object of type file. Set it to your file's path, and then we will pass this fi**le** instance to the Scanner class for scanning. The Scanner class will read the file line-by-line.
- Use the nextLine() method to read a line.
- Split the file by delimiter by using String.split() method.
- After the split, we can store data in ArrayList. We could store that line as a ***String[]*** array as shown below:
    - ArrayList<String[]>
- After that, for display, we can Iterate through Arraylist.

Learning Objective:

After this lab, learners will have demonstrated the ability to read a Delimited File using Java and using java methods.

Example 1

[Click here to Download the Dummy file (Car.csv).]

Remember the path or location of the downloaded file. We will use that file in this Lab.

Create a class named **ScanDelimiterdFile,** or give any name to the class. Write the code below in that class.

Note: Do not forget to change the path or location of the file (cars.csv) at line number 9.

```
  1. import java.io.File;
  2. import java.io.FileNotFoundException;
  3. import java.util.Scanner;
  4. import java.util.ArrayList;
  5. public class ScanDelimiterdFile{
  6.    public static void main(String[] args) throws FileNotFoundException {
  7.
  8.            try {
  9.                String location = "C:/Users/Downloads/cars.csv";
 10.                File file = new File(location);
 11.                Scanner input = new Scanner(file);
 12.                ArrayList<String[]> data = new ArrayList<String[]>();
 13.                while (input.hasNextLine()) {
 14.                    String Line = input.nextLine();
 15.                    String[] splitedLine = Line.split(",");
 16.                    data.add(splitedLine);
 17.                }
 18.                for (String[] line : data) {
 19.              //System.out.println(line[0] + "|" + line[1] + "|" + line[2] + "|" +
line[3]  + line[4] + "|" + line[5] + "|" + line[6] + "|" + line[7] + "|" + line[8]);
 20.                        System.out.println("Car Name :" + line[0] );
 21.                        System.out.println("MPG :" + line[1] );
 22.                        System.out.println("Cylinder :" + line[2] );
 23.                        System.out.println("Displacement :" + line[3]);
 24.                        System.out.println("Horsepower :" + line[4]);
 25.                        System.out.println("Weight :" + line[5]);
 26.                        System.out.println("Acceleration :" + line[6]);
 27.                        System.out.println("Model :" + line[7]);
 28.                        System.out.println("Origin :" + line[8]);
 29.                        System.out.println("===============================");
 30.                    }
 31.
 32.            } catch (FileNotFoundException e) {
 33.                System.out.println("File not found! ");
 34.                e.printStackTrace();
```

```
        }
    }
}
```

The **hasNext()** method verifies whether the file has another line, and the **nextLine()** method reads and returns the next line in the file.

Example 2

Let’s make our code more professional using the concept of “Encapsulation.”

Another way of handling a delimited file is by creating something called a ***Model, Pojo,*** *or* ***Entity***.

A *Model* is simply a class containing variables with **getter()** methods and **setter()** methods, corresponding to each column of the delimited file and containing everything a normal class can contain.

Assume that you have **‘course’** information in the form of a CSV file. As a developer, it is your responsibility to extract data from a file, and then display the data in a console. Finally, you import data into the database. This process is called ETL (Extract Transformation Load). Let’s see first how we can **extract/read** data from a CSV file in a professional way.

Click here - Download the Dummy file (CourseData.csv).

Create a class named **course**, and write the code below in that class. This will be our Model class.

```
public class course {
   private String code, course_name, instructor_name;
   public course (String code, String name, String instructor) {
       this.code = code;
       this.course_name = name;
       this.instructor_name = instructor;
   }
   public course () {
   }
   public String getCode() {
       return code;
   }
   public void setCode(String code) {
       this.code = code;
   }
```

```
    public String getCourse_name() {
        return course_name;
    }

    public void setCourse_name(String course_name) {
        this.course_name = course_name;
    }

    public String getInstructor_name() {
        return instructor_name;
    }

    public void setInstructor_name(String instructor_name) {
        this.instructor_name = instructor_name;
    }
}
```

If you notice, that class has only ***private variables, constructors, getters(), and setters()*** for each variable, so we can say it is Encapsulation.

Create a class named **MyRunner**. Write the below code

💡***Note: Do not forget to change the path or location of the file(*** `CourseData.csv` ***)***.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class MyRunner {

   public static void main(String[] args) throws FileNotFoundException {

       try {
//-----  change file path, as per your file location
  String location = "C:/Users/Downloads/CourseData.csv";
           File file = new File(location);
           Scanner input = new Scanner(file);
           ArrayList<course> data = new ArrayList<course>();

           while (input.hasNextLine()) {
               String Line = input.nextLine();
```

```
                String[] splitedLine = Line.split(",");

// course cObj1 = new course(splitedLine[0], splitedLine[1], splitedLine[2]);
                course cObj = new course();
                cObj.setCode(splitedLine[0]);
                cObj.setCourse_name( splitedLine[1]);
                cObj.setInstructor_name(splitedLine[2]);

                data.add(cObj);
            }

            for (course c : data) {
                System.out.println(c.getCode() + " | " + c.getCourse_name() + "|"
+ c.getInstructor_name());
                System.out.println("================================");
            }

        } catch (FileNotFoundException e) {
            System.out.println("File not found! ");
            e.printStackTrace();
        }
    }
}
```

Output:

```
Course Code | Course Name|Instructor name
================================
CIS135 | Object-Oriented Programming |Michael Gabriel
================================
CIS235 | Object-oriented Programming II|Bairon Vasquez
================================
JIA254 | Java Full Stack|Haseeb
================================
JJA698 | Java Developer with HTML|Jafer
================================
RTP856 | React Developer|James Santana
================================
```

Submission Instructions:
Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: GLAB - 303.13.1 - Reading a Delimited File **Points:** 100 **Assignment Group:** Module 303: Java SE Review (Not Graded) **Display Grade As:** Complete/Incomplete **Do not count this assignment towards the final grade:** Checked **Submission Types:** File Uploads **Everything else is the default.**