EXERCISE 12, WORKING WITH FILES

FUNDAMENTAL

# Exercise 12, Java - Working with Files

## Objective

The objective of this practical session is for you to become familiar with a number of classes found in the java.io namespace and practice reading and writing to files.

## Part 1 – File I/O Practice

1. Back in the **labs** project which you created in *Lab1*, add a new package called **lab12**.
2. Add a new class called **Program** to this package with a *main*() method.
3. Add three pre-supplied text files to your ***src*** folder.   
   **course.txt, trainer.txt and courseTrainer.txt**

**Tip:** to add these right mouse click on the src folder and then select **import** > **general** > **file system**.

1. Create a method in the Program class with following signature

**private** **static** String[] readCSV(String inFile) **throws** IOException {  
}

We've not covered Exception handling so just ignore the   
**throws** IOException part for now!

Please also change the main() as:  
**public** **static** **void** main(String[] args) **throws** Exception {

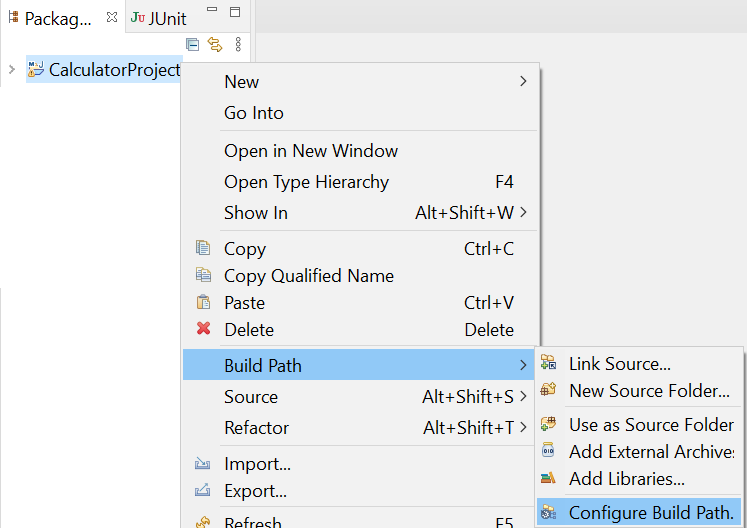
1. Write code in the readCSV method to read one line the inFile parameter and split the resulting String using "," as delimiter. Return the String[ ] returned from the split method.

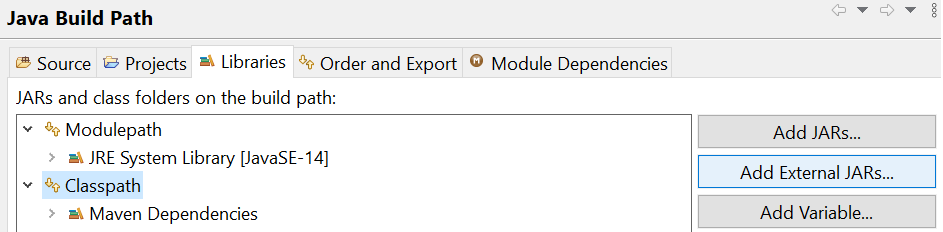
**Tip:** use the String's split(",") method. This method returns an String[ ].

1. Back in main(), call the readCSV() method and pass it the "src/course.txt" as file name and capture the result back in a variable called **courses**.  
      
   Do the same for "src/trainer.txt" and capture the result in a variable called trainers.
2. Create a file called **trainersAndCourses.txt** and append the courses and trainers arrays to this file.
3. Run your code and make sure it works.

**Part 2 – Reading JSON files.**

1. In this part of this exercise you’ll create some code to read a JSON file and parse the text in order to extract an array of customers stored within it.  
   Create a new project (or add a class to an existing Java project) to read and process JSON data.
2. Before you can use the Gson module, you need to import it. Please follow these steps.  
   Right mouse click on your project and then select the "Build Path" > "Configure Build" Path options



1. Select the “Class path” option and then click the “Add External JARs…” button. Select the provided “gson-2.6.2.jar” file in the Windows Explorer.  
   Click the “Apply and Close” button to complete the import.  
     
   

View the provided “customers.json” file in an online JSON viewer (such as <http://jsonviewer.stack.hu/> ) and you’ll see that it contains an array of customer object.  
  
  
  
You’ll soon read the customer data and create an array of customers. Therefore, you’ll need to create a customer class   
 **class** Customer {  
 String CustomerID, CompanyName, ContactName, ContactTitle;  
 String Address, City, Region, PostalCode, Country;  
}  
You would usually declare these fields as **private** and then use your IDE to create getters and setters for the private fields but we’ll leave these as internal to the package to concentrate on the main subject.  
Please copy the following code to your main() method or create another method to house it.  
  
**public** **static** **void** main(String[] args) **throws** Exception{  
 Gson gson = **new** Gson();  
 String fileName = "C:\\qa\\customers.json";  
 String content = **new** String(Files.*readAllBytes*(Paths.*get*(fileName)));  
 Customer[] customers = gson.fromJson(content, Customer[].**class**);  
 System.***out***.println(customers[0].CustomerID);  
}  
  
The above code prints the first customer’s **CustomerID**.  
Please note, the spelling of this field is not accidental but matches the text in the **customers.json** file.Please create a loop to print more of the data.  
  
Run the code to test it.

Writing objects as JSON:  
How would you write a single object to a file?   
How about a List of objects?   
Here is a sample code to get you started:  
   
 Gson gson = **new** Gson();

**int**[] nos = {1,3,5,7,9};

String res = gson.toJson(nos);

System.***out***.println(res);

String[] names= {"Bob","Fred","Wilma"};

res = gson.toJson(names);

System.***out***.println(res);

Student[] students = {  
 **new** Student("Bob",21),   
 **new** Student("Fred",32),   
 **new** Student("Wilma",26)

};

res = gson.toJson(students);

System.***out***.println(res);

//-----------------------------------

**class** Student{

String name;

**int** age;

ArrayList<String> subjects = **new** ArrayList<>();

**public** Student(String name, **int** age) {

**this**.name = name; **this**.age = age;

subjects.add("Java");

subjects.add("C#");

subjects.add("Network");

}

}

## Part 3 – Relate courses and trainers (optional and only if you've time)

Add a pre-supplied text file called **trainerCourse.txt** to the **src** folder. This file contains rows of two **","** separated integers.  
The two integers point to trainer numbers (the first value) and course titles (second number).   
Trainer and course numbers being the order in which trainers and courses are stored in the files.  
For example, **1,2** indicates   
"*Mike Baradaran*" (**1st** row of trainer.txt), teaching  
"*Systems Development Basics*" (**2nd** row of course.txt).

Convert all the numbers to trainer names and course titles in order to produce a report.

**\*\* End \*\***