CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Web Services
420-941-VA

Teacher: Samir Chebbine **Lab 3** **Oct 07, 2024**

Lab 3: Web Services using REST Implementation

Complete all these following programs in class**.** All *missing coding statements* are presented during class time and in Presentation 3.

Create and Submit a Word file ***Lab3WebServicesYourName.doc*** which contains Answers of theory questions if any and output screenshots for every Java EE Project. Submit the Java projects too and submit the whole Lab 3 as compressed zip file

1. **Creating Dynamic Web Project and Convert it into Maven Project**
   a) Create a new Dynamic Web project called **WebHelloRESTProject and convert it into Maven Project**.
   b) Add **Maven Project dependencies** as stated in my Lab 3 in **pom.xml**. Create new package called **webHelloREST** as shown in Figure 1. Notice the REST URL mapping ("HiHi") used in Java annotation @Path.
   c) Deploy **WebHelloRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**

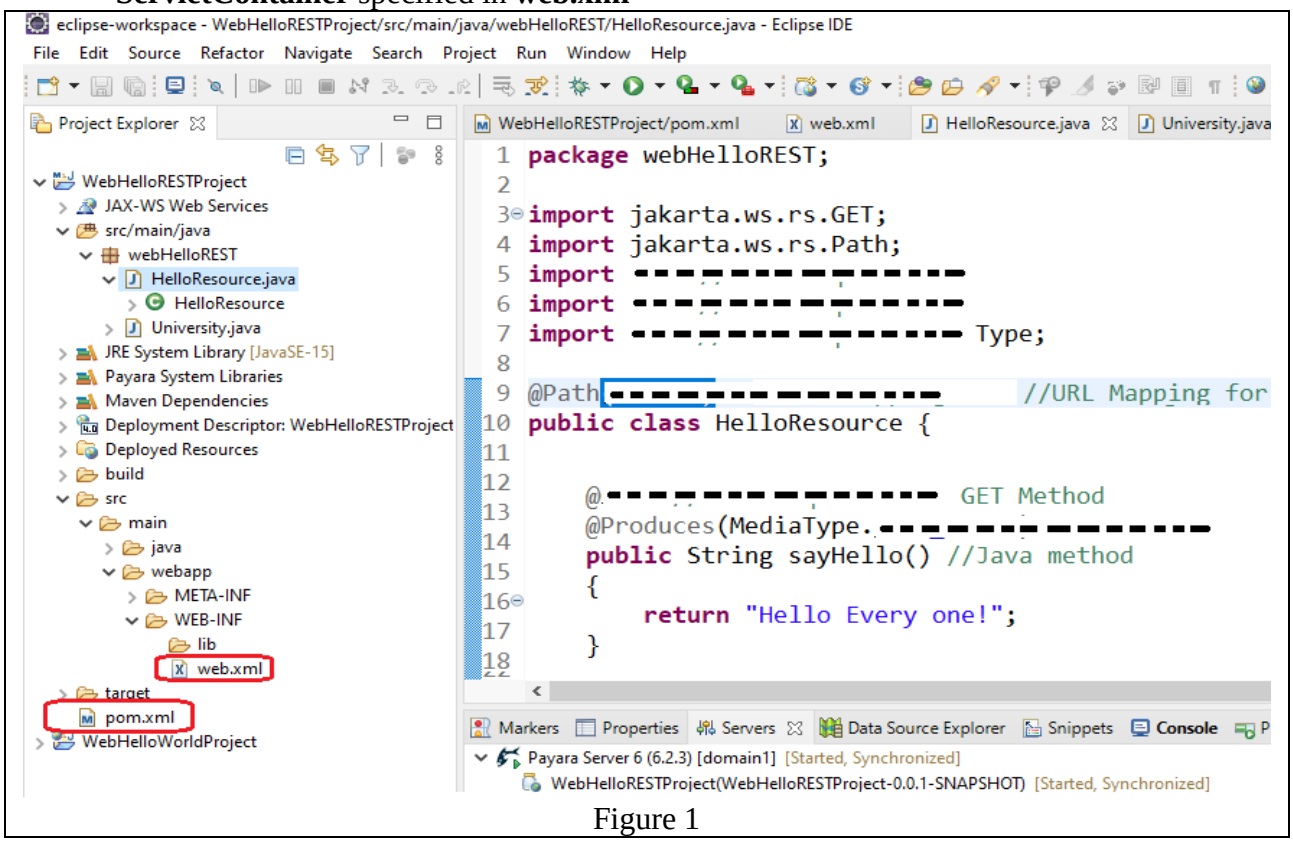


Figure 1

   d) Using Postman, display screenshots testing to each media type included in my Lab3 related to plain text, HTML, JSON output as shown in Figure 2.

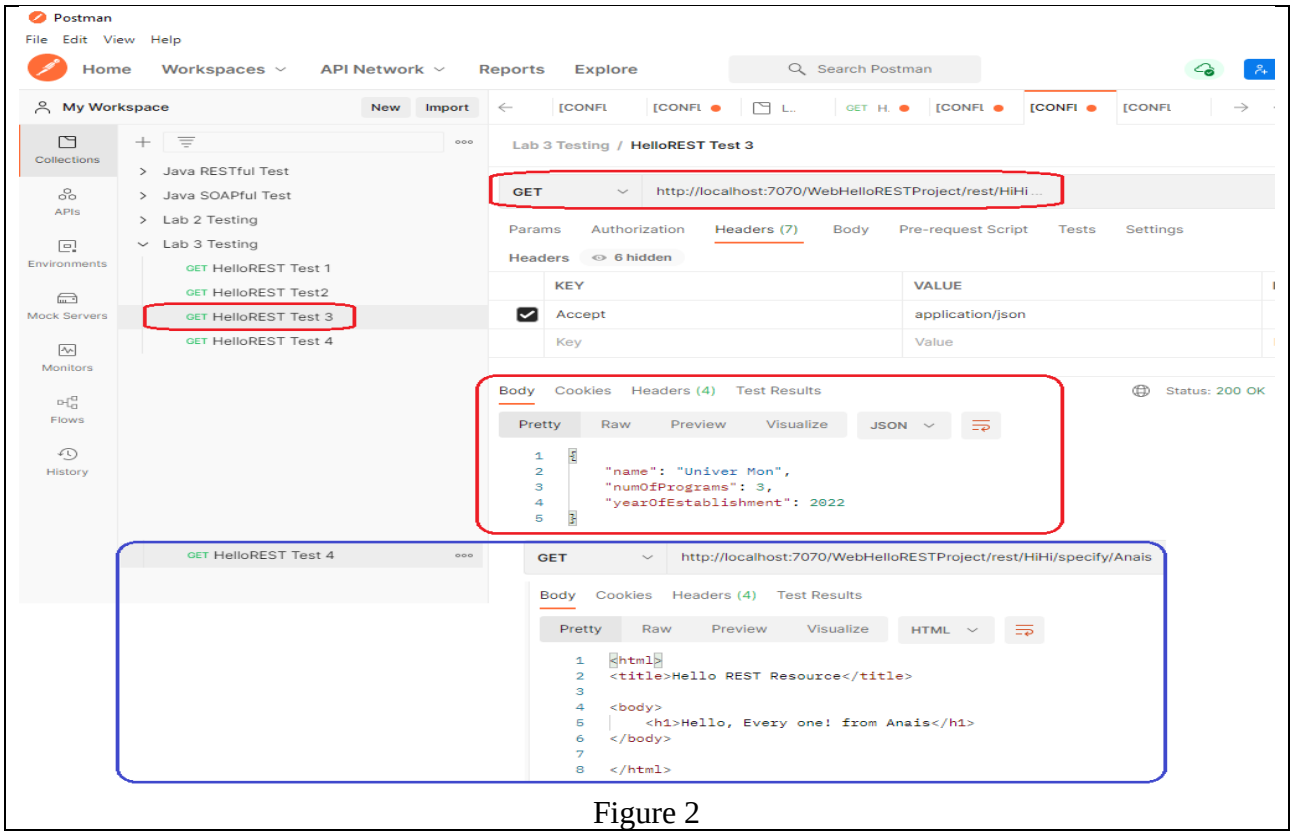


Figure 2

2. **Maven Dynamic Web Project: WebMathOperationsRESTProject**
   a) Create a new Dynamic Web project **WebMathOperationsRESTProject and convert it into Maven Project**. Check the output using Postman. Save your own screenshots.
   b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **mathOperationsREST.**
   c) Deploy **WebMathOperationsRESTProject** within GalssFish Server.
   d) You need to develop a **Java class** called `MathOp`, which takes `x, y, z` as **private** `non static` members. The `MathOp` class contains the following method members:
      - Add a method called `calculateSum()` in `MathOp` class that returns `(x+2*y+3*z).`
      - Add a method called `calculatePrd()` in `MathOp` class that returns `(x*2*y*3*z).`
   e) Create a new REST Resource class **WebMathResource.java**

1. Add a path URL mapping **("MathOp**") to access REST resource using appropriate REST annotation and call the following methods **calculateHTMLOp()/displayXYZJSON()**.
2. Add a method **calculateHTMLOp()** that returns a HTML media type using appropriate Java REST annotations.

Add appropriate statements in **calculateHTMLOp()** using **query string parameters** `x, y, z` that calls implemented methods `calculateSum()/calculatePrd()` in `MathOp`.

3. Add a method **displayXYZJSON()** that returns a JSON media type and instantiate an `object of MatOp class type`. Set its data attributes to (1, 2, 3).
4. Add a new path URL mapping ("/listArray…") that calls a method **displayListZYZ()** that returns a HTML media type using appropriate Java REST annotations.

-Add appropriate statements in **displayListZYZ ()** to instantiate a Java data structure **Array List** `of object of MatOp class type` to be referenced by (`listXYZ`). Add every component of Array List course object to the following values `(1,2,3)(4,5,6)(7,8,9).`

Skip through `Array List of object` (`listXYZ`) and display its components as shown hereafter.

5. Add a new path URL mapping ("/OpHashMap…") with path parameter x as search parameter to access REST resource searching into Hash Map.

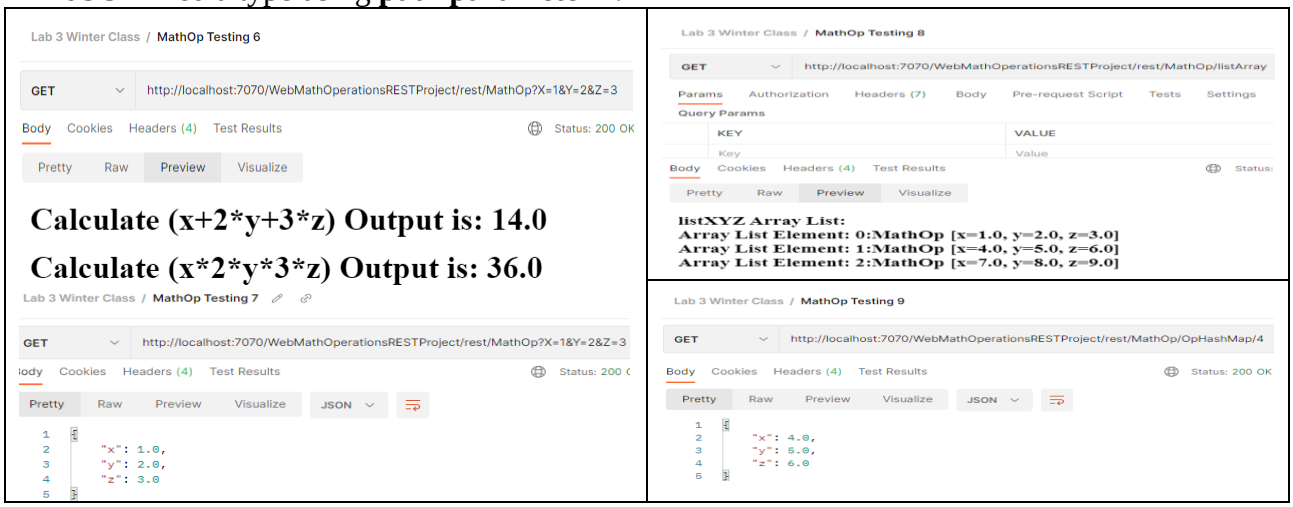
-Add a method **searchHashMapListZYZ()** using **path parameter** `x` and returns a JSON

media type using appropriate Java REST annotations and will be fired upon using URL mapping ("/OpHashMap…").

-Add appropriate statements in **searchHashMapListZYZ ()** to instantiate a data structure **HashMap** `of MathOp class type` to be referenced by (`opHashMap`) where hash map key represents `x (path parameter)` and value hash map of `MathOp class type`. Set every component of hash map to the following values:

`x = 1,y = 2,z = 3/ x = 4,y = 5,z = 6 / x = 7,y = 8,z = 9`

-Skip through `Hash Map collection` (`opHashMap`) and display the result of Hash Map search in JSON media type using **path parameter** `x`.



3. Maven Dynamic Web Project: WebCarRESTProject

a) Create a new Dynamic Web project called **WebCarRESTProject and convert it into Maven Project**.

b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **webCarREST**.

c) Deploy **WebCarRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**

d) You need to develop a **Java class** called **`Car (see Block3)`**, which takes `vin, desc, price` as **private** `non static` members. The `Car` class contains the following method members:

- Add a method called `discountPrice()` in Car class to calculate price discount of a given car after applying 10% discount on car price.

e) Create a new REST Resource class **WebCarResource.java**

1. Add a path URL mapping ("WebCar") to access REST resource using appropriate Java REST annotation.

2. Add a method **displayHTMLCarInfo()** that returns a HTML media type using appropriate Java REST annotations.

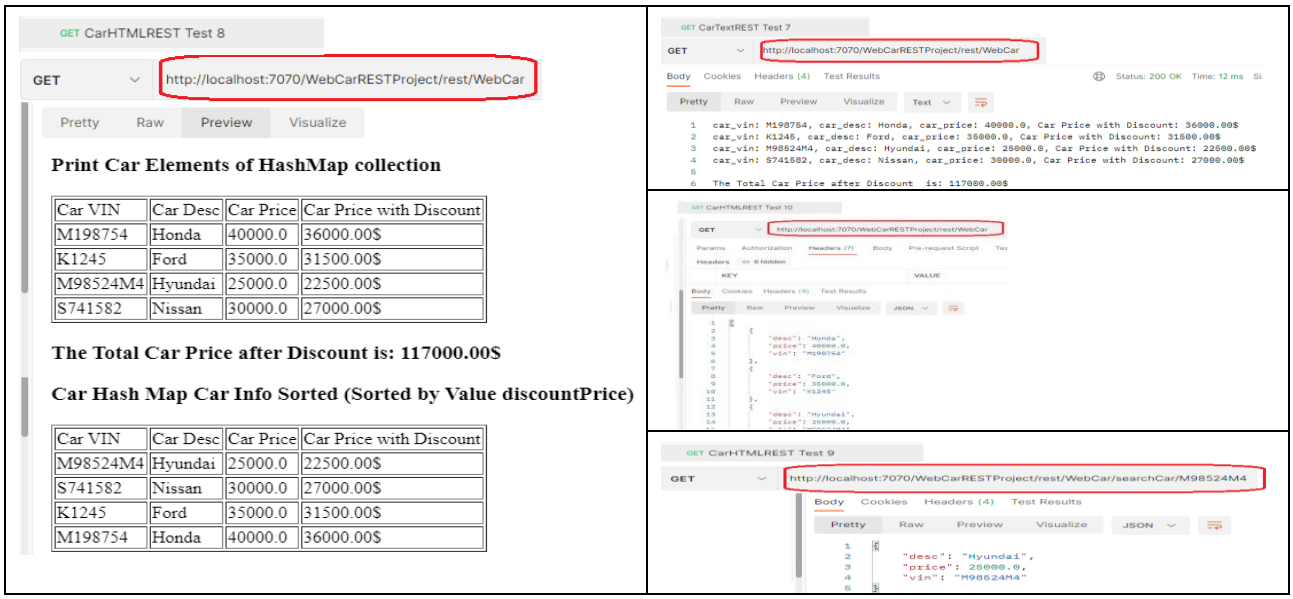
3. Add appropriate statements in **displayHTMLCarInfo()** to instantiate a data structure **HashMap** `of Car class type` to be referenced by (`carHashMap`) where hash map key represents `vin` and value hash map of `Car class type`. Set every component of hash map to the following values read from text file **Car.in (use tab as separator)**:

`vin = K1245, desc = Ford, price =35000/vin =M198754, desc = Honda, price =40000`
`vin =M98524M4,desc = Hyundai,price =25000/vin =S741582, desc = Nissan price =30000`

4. Skip through `Hash Map collection` (`carHashMap`) and display its unsorted components and sorted components with respect to car price discount into web table respectively as shown hereafter **`(see Block3 for sorting)`**. Check output using Postman. Save your own screenshot.

5. Add a method **displayTextCarInfo ()** that returns the same output as plain TEXT media type using appropriate Java REST annotations. Check the output using Postman.

6. Add a method **displayJSONCarInfo ()** that returns elements of `Hash Map collection` (`carHashMap`) as JSON media type. Check the output using Postman.
7. Add a new path URL mapping ("/searchCar…") with path parameter vin as search string parameter to access REST resource searching into Hash Map. Save your own Postman screenshot in word document.
8. Add a method **searchJSONCarInfo(String car_vin)** that returns the result of Hash Map search in JSON media type and will be fired upon using URL mapping ("/searchCar…") as shown hereafter.



4. Maven Dynamic Web Project: WebBillingRESTProject

a) Create a new Dynamic Web project called **WebBillingRESTProject and convert it into Maven Project**.
b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **webBillingREST**.
c) Deploy **WebBillingRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**

d) You need to develop a **Java class** called **`Billing`**, which takes `client_ID, client_LName, and client_FName, product_Name, prd_Price=0, prd_Qty` as **private** `non static` members. The variables called `Fed_Tax`, `Prv_Tax` as **public** and `static` data members. The `Billing` class contains the following method members:
(Notice that it is same class as in Lab 1; however, you need to add data attribute `client_ID`)

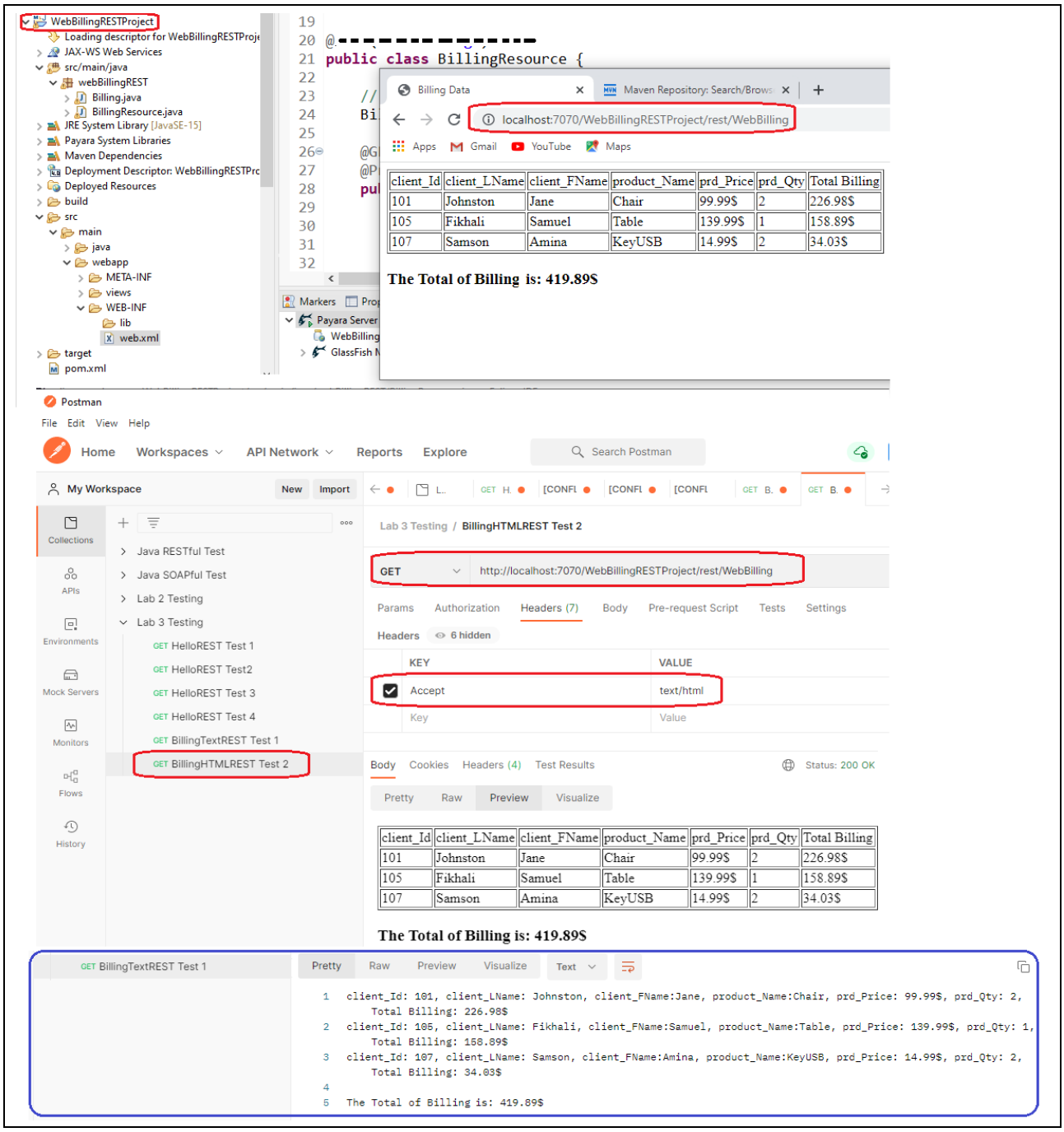
- Add default constructor `(client_ID, client_LName, client_FName, product_Name prd_Price=0, prd_Qty=0)` and constructor with parameters within the Billing class in order to initialize the data members of every object.
- Add public setter methods and getter methods `(setClient_LName()…,getClient_LName()…,)` in Billing class to modify the values of private members.
- Add a method called `CalculateBilling()` in Billing class to calculate the total of billing

```
T_Billing = (prd_Price* prd_Qty) + (prd_Price*prd_Qty)* Fed_Tax + (prd_Price*prd_Qty)* Prv_Tax
```

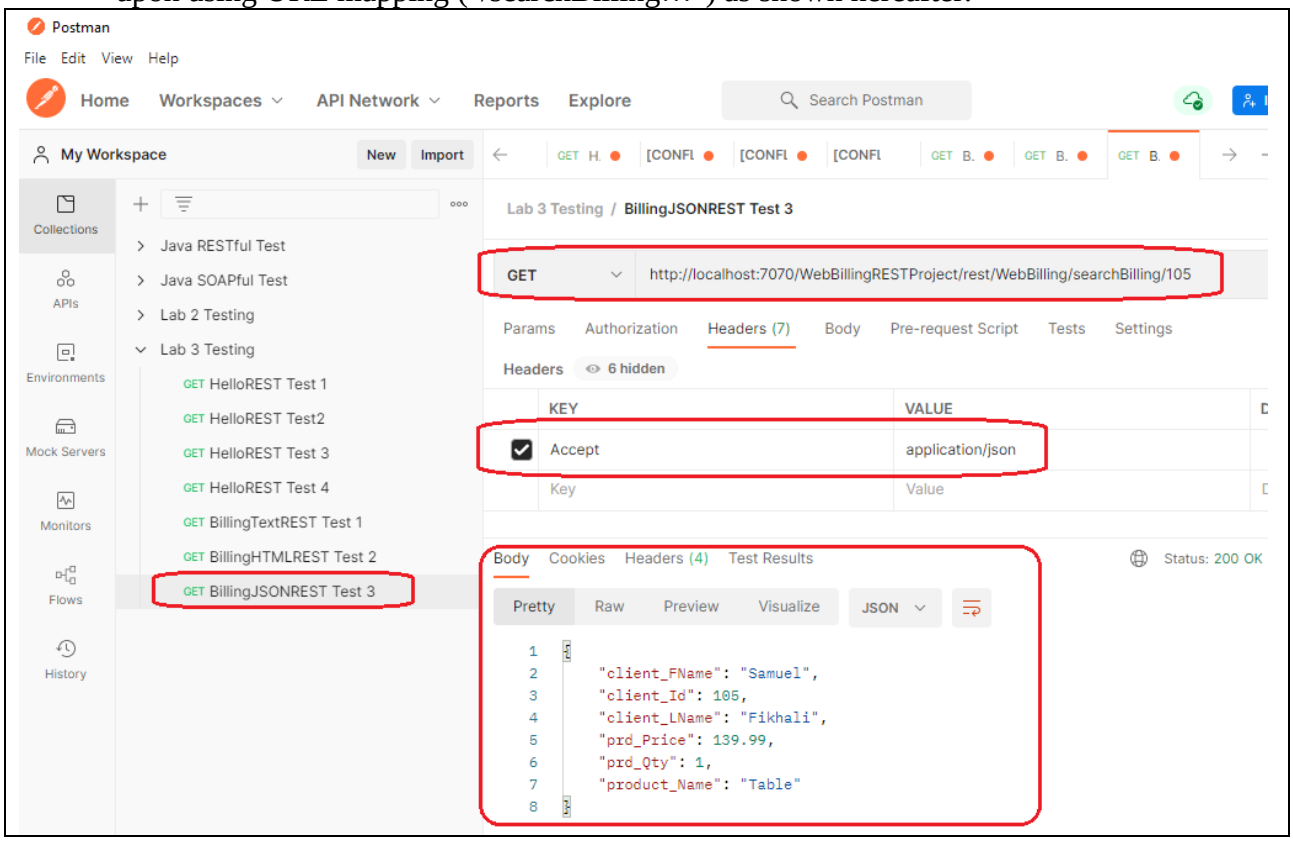
e) Create a new REST Resource class **WebBillingResource.java**
1. Add a path URL mapping ("WebBilling") to access REST resource using appropriate Java REST annotation.

2. Add a method **displayHTMLBillingInfo()** that returns a HTML media type using appropriate Java REST annotations.
3. Add appropriate statements in **displayHTMLBillingInfo()** to instantiate a Java data structure **Array List** `of object of Billing class type` to be referenced by (`BillingList`). Add every component of Array List Billing object using the implemented setter methods (`setClient_ID(), setClient_LName(),setClient_FName (),setproduct_Name(), setPrd_Price(), setPrd_Qty()`) to the following values read from text file **Billing.in (use tab as separator)** (`Fed_Tax=0.075 Prv_Tax= 0.06`)

4. Skip through `Array List of object` (`BillingList`) and display its components into Web Table as shown hereafter. Check the output using Postman. Save your own screenshot.
5. Add a method **displayTextBillingInfo()** that returns the same output as plain TEXT media type using appropriate Java REST annotations. Check the output using Postman.



6. Add a new path URL mapping ("/searchBilling…") with path parameter client_id as search integer parameter to access REST resource searching into Array List using appropriate Java REST annotation. Save your own Postman screenshot in word document.
7. Add a method **searchJSONBillingInfo(int client_id)** that returns the result of Array List search in JSON media type using appropriate Java REST annotations and will be fired upon using URL mapping ("/searchBilling…") as shown hereafter.



5. Maven Dynamic Web Project: WebCourseRESTProject

a) Create a new Dynamic Web project called **WebCourseRESTProject and convert it into Maven Project**.
b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **webCourseREST**.
c) Deploy **WebCourseRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**
d) You need to develop a **Java class** called `Course` that *represents* a template of the fields used in defining the columns of a given table *Course* which takes `course_no`, `course_name`, `max_enrl` as **private** `non static` data members. `credits` as **public** and `static` data member. The `Course` class contains the following method members:
(Notice that it is same class as in Lab 1)

- Add **constructor with parameters** within the `Course` class to initialize the **private** data members `(course_no, course_name, max_enrl, credits)` of every object.
- Add public **Mutator** (**setter**) methods in `Course` class to modify the values of private members.
- Add public **Accessor** (**getter**) methods in `Course` class to read the values of private members.
- Add a return method called `CalculateTotalFees()`in `Course` class to return the total fees of all enrolled students according to the following formula `max_enrl*250`.

e) Create a new REST Resource class **WebCourseResource.java**

1. Add a path URL mapping ("WebCourse") to access REST resource using appropriate Java REST annotation.
2. Add a method **displayHTMLCourseInfo()** that returns a HTML media type using appropriate Java REST annotations.
3. Add appropriate statements in **displayHTMLCourseInfo()** to instantiate a Java data structure **Array List** `of object of Course class type` to be referenced by (`CourseList`). Add every component of Array List course object to the following values read from text file **Course.in (use tab as separator)**.
4. Skip through `Array List of object` (`CourseList`) and display its components into Web Table as shown hereafter. Check the output using Postman. Save your own screenshot.
5. Add a method **displayTextCourseInfo()** that returns the same output as plain TEXT media type using appropriate Java REST annotations. Check the output using Postman.
6. Add a new path URL mapping ("/searchCourse…") with path parameter course_no as search String parameter to access REST resource searching into Array List using appropriate Java REST annotation. Save your own Postman screenshot in word document.
7. Add a method **searchJSONCourseInfo(String course_no)** that returns the result of Array List search in JSON media type using appropriate Java REST annotations and will be fired upon using URL mapping ("/searchCourse…") as shown hereafter.