

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

- Create a new Dynamic Web project called **WebHelloRESTProject** and convert it into **Maven Project**.
- 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**.
- Deploy **WebHelloRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**

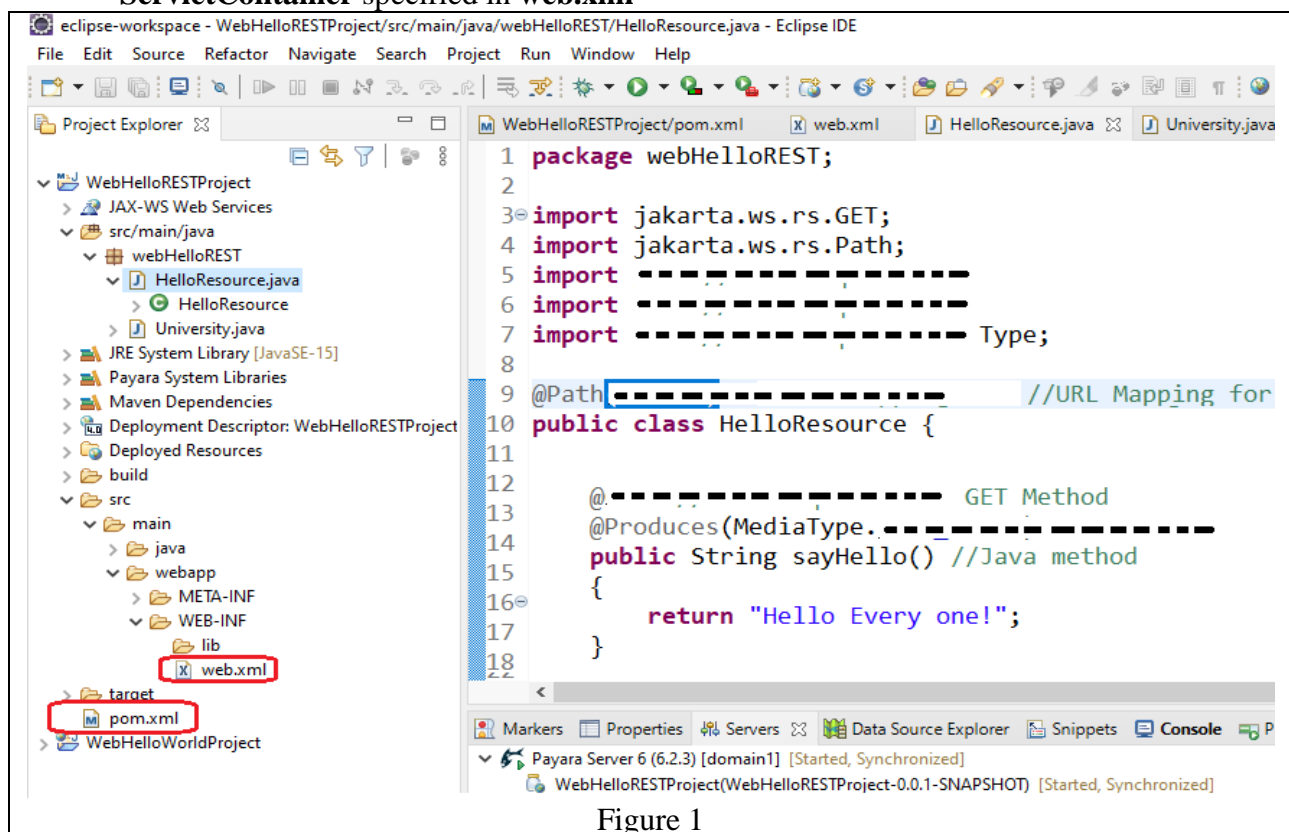


Figure 1

- 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.

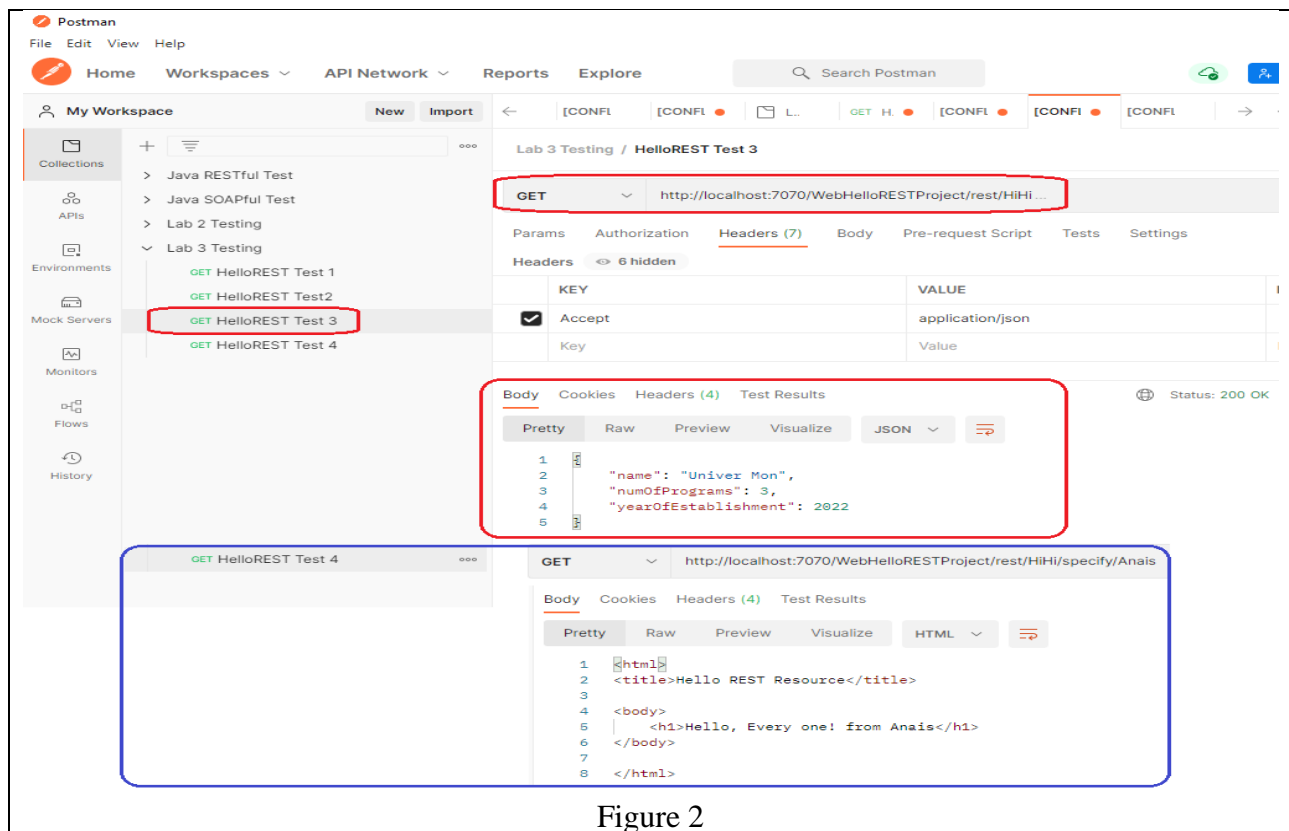


Figure 2

2. Maven Dynamic Web Project: WebMathOperationsRESTProject

- Create a new Dynamic Web project **WebMathOperationsRESTProject** and convert it into **Maven Project**. Check the output using Postman. Save your own screenshots.
- Add **Maven Project dependencies** in **pom.xml**. Create new package called **mathOperationsREST**.
- Deploy **WebMathOperationsRESTProject** within GalssFish Server.
- 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 $(x*2*y*3*z)$.
- Create a new REST Resource class **WebMathResource.java**
 - Add a path URL mapping ("**MathOp**") to access REST resource using appropriate REST annotation and call the following methods **calculateHTMLOp()/displayXYZJSON()**.
 - 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**.
 - 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).
 - 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.
 - 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**.

Lab 3 Winter Class / MathOp Testing 6

GET http://localhost:7070/WebMathOperationsRESTProject/rest/MathOp?X=1&Y=2&Z=3

Body Cookies Headers (4) Test Results Status: 200 OK

Calculate (x+2*y+3*z) Output is: 14.0
Calculate (x*2*y*3*z) Output is: 36.0

Lab 3 Winter Class / MathOp Testing 7

GET http://localhost:7070/WebMathOperationsRESTProject/rest/MathOp?X=1&Y=2&Z=3

Body Cookies Headers (4) Test Results Status: 200 OK

```
1 {"x": 1.0,
2  "y": 2.0,
3  "z": 3.0}
```

Lab 3 Winter Class / MathOp Testing 8

GET http://localhost:7070/WebMathOperationsRESTProject/rest/MathOp/listArray

Body Cookies Headers (4) Test Results Status: 200 OK

listXYZ Array List:
Array List Element: 0:MathOp [x=1.0, y=2.0, z=3.0]
Array List Element: 1:MathOp [x=4.0, y=5.0, z=6.0]
Array List Element: 2:MathOp [x=7.0, y=8.0, z=9.0]

Lab 3 Winter Class / MathOp Testing 9

GET http://localhost:7070/WebMathOperationsRESTProject/rest/MathOp/OpHashMap/4

Body Cookies Headers (4) Test Results Status: 200 OK

```
1 {"x": 4.0,
2  "y": 5.0,
3  "z": 6.0}
```

3. Maven Dynamic Web Project: WebCarRESTProject

- Create a new Dynamic Web project called **WebCarRESTProject** and convert it into **Maven Project**.
- Add **Maven Project dependencies** in **pom.xml**. Create new package called **webCarREST**.
- Deploy **WebCarRESTProject** within GalssFish Server to be executed by **Servlet class ServletContainer** specified in **web.xml**
- 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.
- Create a new REST Resource class **WebCarResource.java**
 - Add a path URL mapping ("WebCar") to access REST resource using appropriate Java REST annotation.
 - Add a method **displayHTMLCarInfo()** that returns a HTML media type using appropriate Java REST annotations.
 - 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
 - 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.
 - 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.

GET CarHTMLREST Test 8

GET <http://localhost:7070/WebCarRESTProject/rest/WebCar>

Print Car Elements of HashMap collection

Car VIN	Car Desc	Car Price	Car Price with Discount
M198754	Honda	40000.0	36000.00\$
K1245	Ford	35000.0	31500.00\$
M98524M4	Hyundai	25000.0	22500.00\$
S741582	Nissan	30000.0	27000.00\$

The Total Car Price after Discount is: 117000.00\$

Car Hash Map Car Info Sorted (Sorted by Value discountPrice)

Car VIN	Car Desc	Car Price	Car Price with Discount
M98524M4	Hyundai	25000.0	22500.00\$
S741582	Nissan	30000.0	27000.00\$
K1245	Ford	35000.0	31500.00\$
M198754	Honda	40000.0	36000.00\$

GET CarTextREST Test 7

GET <http://localhost:7070/WebCarRESTProject/rest/WebCar>

Body Cookies Headers (4) Test Results Status: 200 OK Time: 12 ms SI

```

1 car_vin: M198754, car_desc: Honda, car_price: 40000.0, Car Price with Discount: 36000.00$
2 car_vin: K1245, car_desc: Ford, car_price: 35000.0, Car Price with Discount: 31500.00$
3 car_vin: M98524M4, car_desc: Hyundai, car_price: 25000.0, Car Price with Discount: 22500.00$
4 car_vin: S741582, car_desc: Nissan, car_price: 30000.0, Car Price with Discount: 27000.00$
5
6 The Total Car Price after Discount is: 117000.00$

```

GET CarHTMLREST Test 10

GET <http://localhost:7070/WebCarRESTProject/rest/WebCar>

Body Cookies Headers (4) Test Results

```

1 {
2   "desc": "Honda",
3   "price": 40000.0,
4   "vin": "M198754"
5 }
6
7 {
8   "desc": "Ford",
9   "price": 35000.0,
10  "vin": "K1245"
11 }
12
13 {
14   "desc": "Hyundai",
15   "price": 25000.0,
16   "vin": "M98524M4"
17 }
18
19 {
20   "desc": "Nissan",
21   "price": 30000.0,
22   "vin": "S741582"
23 }
24

```

GET CarHTMLREST Test 9

GET <http://localhost:7070/WebCarRESTProject/rest/WebCar/searchCar/M98524M4>

Body Cookies Headers (4) Test Results

```

1 {
2   "desc": "Hyundai",
3   "price": 25000.0,
4   "vin": "M98524M4"
5 }

```

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.

The screenshot displays an IDE (Eclipse) and Postman. In the IDE, the `BillingResource` class is shown with the `displayHTMLBillingInfo()` method. The method uses `@GetMapping` and `response.setContentType("text/html")` to return an HTML table of billing data and a total. The table data is as follows:

client_Id	client_LName	client_FName	product_Name	prd_Price	prd_Qty	Total Billing
101	Johnston	Jane	Chair	99.99\$	2	226.98\$
105	Fikhali	Samuel	Table	139.99\$	1	158.89\$
107	Samson	Amina	KeyUSB	14.99\$	2	34.03\$

The total billing is 419.89\$.

Postman shows a GET request to `http://localhost:7070/WebBillingRESTProject/rest/WebBilling` with the `Accept` header set to `text/html`. The response is a 200 OK status, displaying the same HTML table and total as the IDE.

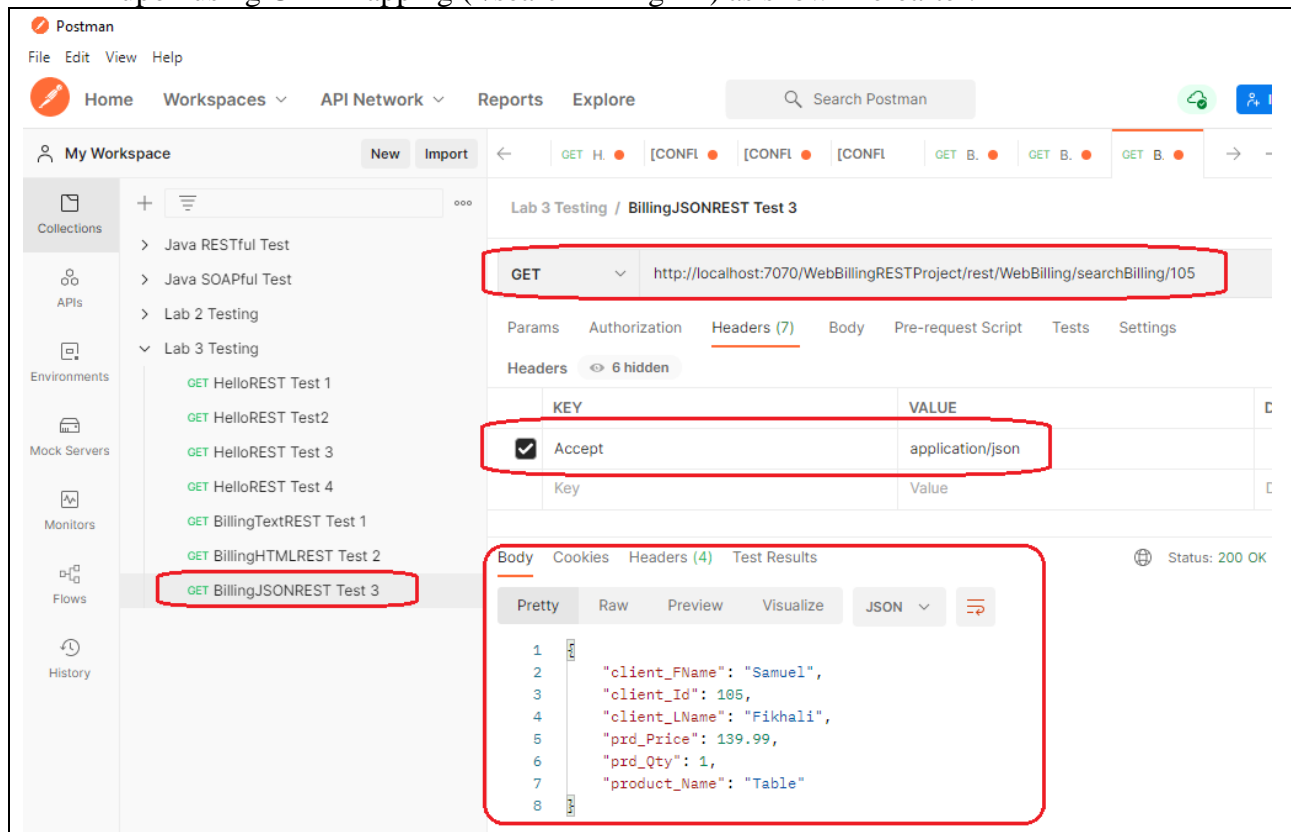
Below the Postman response, the raw text output of the `displayTextBillingInfo()` method is shown:

```

1 client_Id: 101, client_LName: Johnston, client_FName:Jane, product_Name:Chair, prd_Price: 99.99$, prd_Qty: 2,
  Total Billing: 226.98$
2 client_Id: 105, client_LName: Fikhali, client_FName:Samuel, product_Name:Table, prd_Price: 139.99$, prd_Qty: 1,
  Total Billing: 158.89$
3 client_Id: 107, client_LName: Samson, client_FName:Amina, product_Name:KeyUSB, prd_Price: 14.99$, prd_Qty: 2,
  Total Billing: 34.03$
4
5 The Total of Billing is: 419.89$

```


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 $\text{max_enrl} \times 250$.

- 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.

Lab 3 Testing / CourseHTMLREST Test 5

GET `http://localhost:7070/WebCourseRESTProject/rest/WebCourse`

Headers (7):

KEY	VALUE
<input checked="" type="checkbox"/> Accept	text/html

Body: Pretty Raw Preview Visualize

Course Number	Course Name	Max Enrolment	Credits	Total Course Fees
MIS 101	Intro. to Info. Systems	140	3	35000.00\$
MIS 301	Systems Analysis	35	3	8750.00\$
MIS 441	Database Management	12	3	3000.00\$
CS 155	Programming in C++	90	3	22500.00\$
MIS 451	Web-Based Systems	30	3	7500.00\$
MIS 551	Advanced Web	30	3	7500.00\$
MIS 651	Advanced Java	30	3	7500.00\$

The Total of Course Fees is: 91750.00\$

GET CourseJSONREST Test 6

GET `http://localhost:7070/WebCourseRESTProject/rest/WebCourse/searchCourse/MIS 441`

Headers (7):

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/json	

Body: Pretty Raw Preview Visualize JSON

```

1  {
2    "course_name": "Database Management",
3    "course_no": "MIS 441",
4    "max_enrl": 12
5  }

```