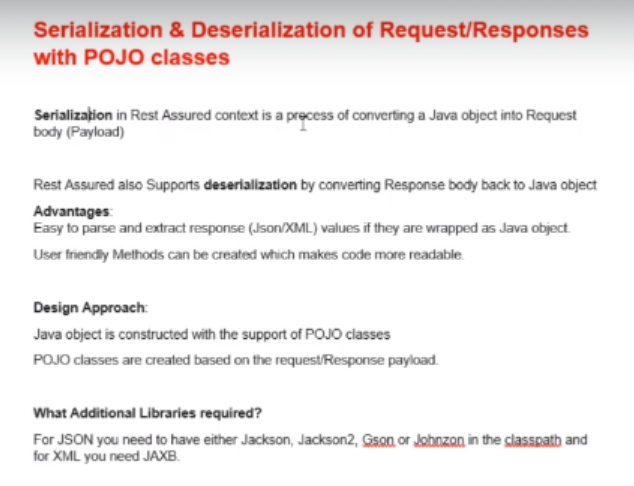
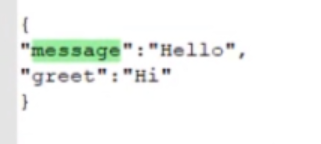
POJO : Plain object java classes



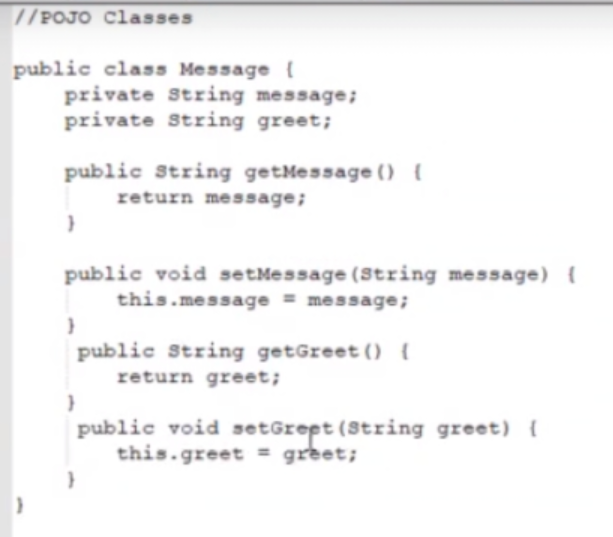
For serialization and deserialization means for request and response, we create separate POJO classes. Because in request we have different fields in Json payload and in response, we receive different json body with different fields in it.

**SERIALIZATION :** Here we use setter method to set values for JSON fields

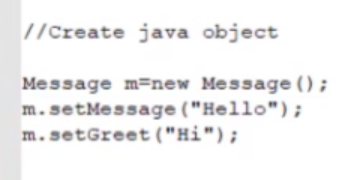
Sample JSON :



We create a Java class and in class we define each fields as Private member and we write public getter and setter method to access the fields. We created a class and methods as shown below :

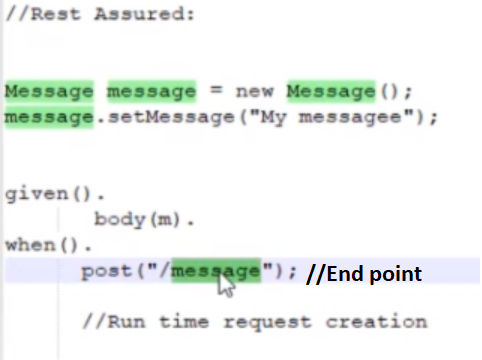


Then we create object of the class and assign values to JSON fields



And this is how we send class object to RestAssured : Rest assured takes care of all rest of the things.

We have to write a class where we define private variables as per Json Payload, and in the same class we maintain getter setter methods, In another, different class, we write a json request where we 1st initialize all variables of pojo class using setter methods (we create object of pojo class and then initialize using setter methods), and then we pass pojo class object instead of sending json payload In request.



**DE-SERIALIZATION :**

Here we convert response to java object. It is exactly same as serialization implementation. Here we use getter method to retrieve values from JSON response.

1. To generate getter and setter methods :

Select private variable > ctrl+shift+s > get getter setter method. Here eclipse will generate getter and setter automatically.

1. For nested Json check that in <https://jsoneditoronline.org/>, if it shows [] means it is array and then we have to create a return type as a list for that node. All nested Jsons are not always indicated with array. But for every nested json we have to create a new class. {**during interviews if json is given and asked to find specific node value, then open above site and click on required node, path will be auto generated at top**.}

GetCourse gc =*given*().queryParam("access\_token",accessToken).expect().defaultParser(Parser.***JSON***)

// here we collect response in object gc of class GetCourse(POJO class)

// defaultParser(Parser.JSON): we tell to scan the response as JSON.

.when()

.get("https://rahulshettyacademy.com/getCourse.php").as(GetCourse.**class**);

//above line as(GetCourse.**class)** tells that we want to get response in to java object of GetCourse class

System.***out***.println(gc.getLinkedin());

System.***out***.println(gc.getInstructor());

// get 1st course title of api json System.***out***.println(gc.getCourses().getApi().get(1).getCourseTitle());

//find price if the course title is : SoapUI Webservices testing

List<Api> apiCourses = gc.getCourses().getApi();

**for**(**int** i=0;i<apiCourses.size();i++)

{

**if**(apiCourses.get(i).getCourseTitle().equalsIgnoreCase("SoapUI Webservices testing"))

{

System.***out***.println(apiCourses.get(i).getPrice());

}

}

Below mentioned JSON looks like this in <https://jsoneditoronline.org/>, Here **Course** is an array so we create separate class for it

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | object |  | {6} | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | instructor | : | RahulShetty | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | url | : | rahulshettyacademy.com | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | services | : | projectSupport | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | expertise | : | Automation | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | courses |  | {3}:- it is an array with 3 elements in it | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | webAutomation |  | [3]:- it is a sub array with 3 elements in it | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 0 |  | {2} | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 1 |  | {2} | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 2 |  | {2} | |
|  |  |  |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | api |  | [2]:-it is a sub array with 2 elements in it | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 0 |  | {2} | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 1 |  | {2} | |
|  |  |  |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | mobile |  | [1]:- it is a sub array with 1 element in it | |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | 0 |  | {2} | |
|  |  |  |
|  |  |  |
|  |  | |  |  |  |  | | --- | --- | --- | --- | |  | linkedin | : | https://url.com | |

**NOTE** : for every array, we create a different, separate class.

GetCourse Json :

{

"instructor": "RahulShetty",

"url": "rahulshettyacademy.com",

"services": "projectSupport",

"expertise": "Automation",

"courses": { //if it is {} and posses elements inside elements create another class

"webAutomation": [//if it is [] and direct elements in it create List<>

{

"courseTitle": "Selenium Webdriver Java",

"Price": "50"

},

{

"courseTitle": "cyprus",

"Price": "40"

},

{

"courseTitle": "Protractor",

"Price": "40"

}

],

"api": [

{

"courseTitle": "Rest",

"Price": "40"

},

{

"courseTitle": "SoapUI",

"Price": "40"

}

],

"mobile": [

{

"courseTitle": "mobile auto",

"Price": "50"

}

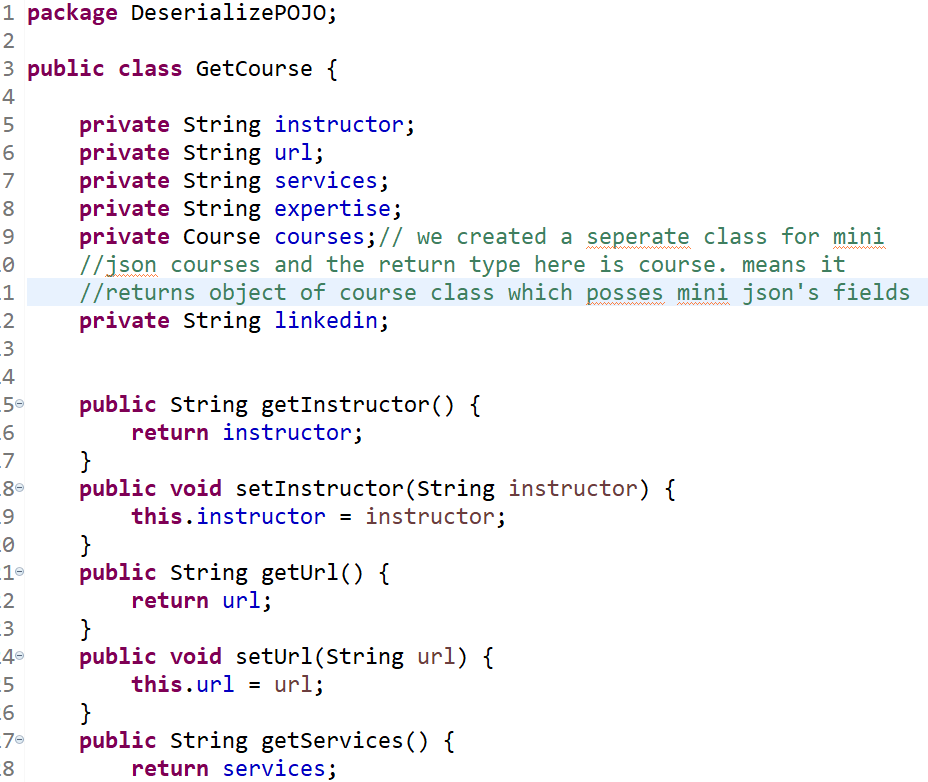
]

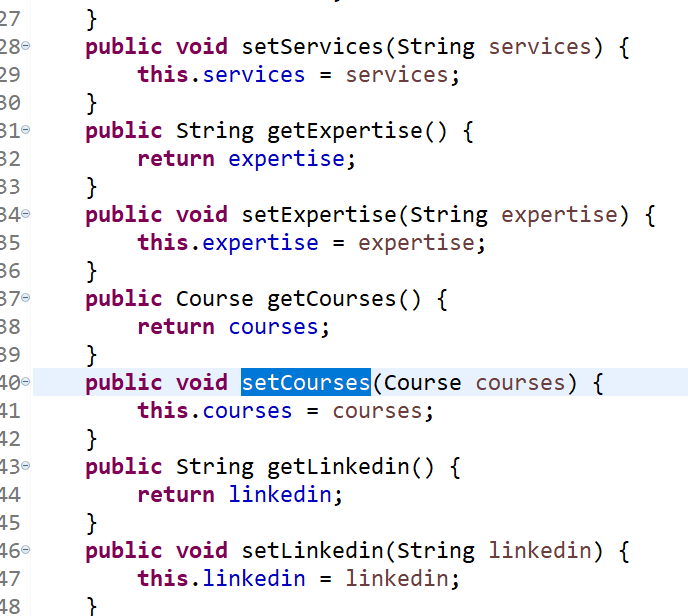
},

"linkedin": "https://url.com"

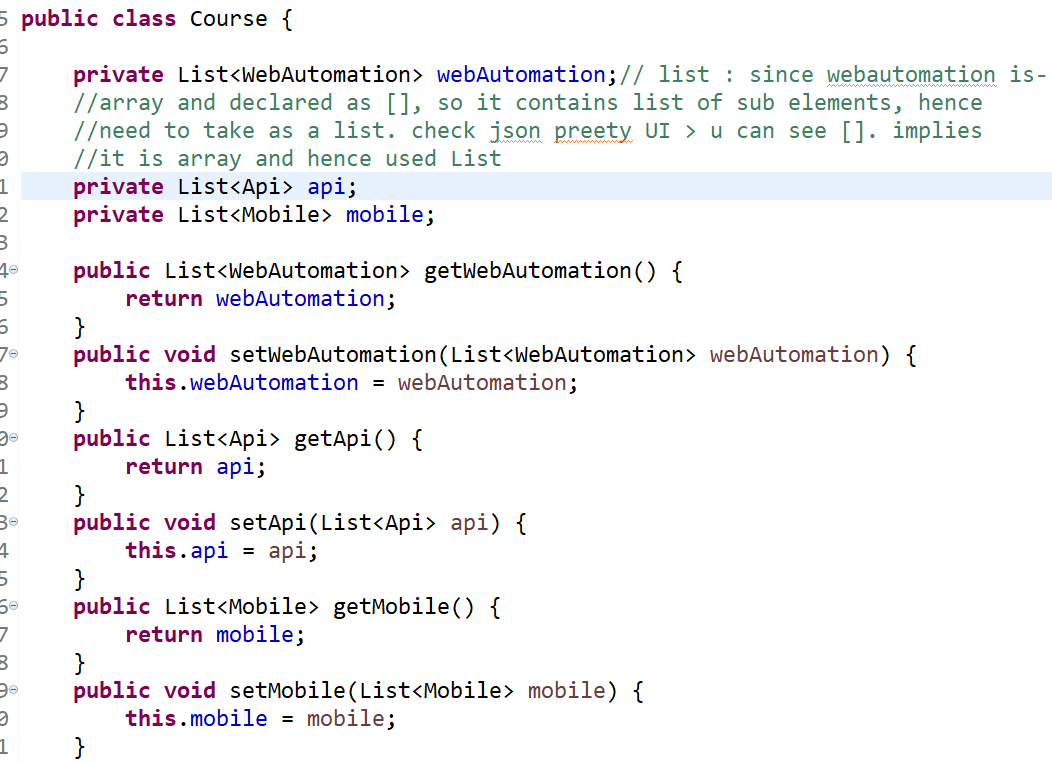
}

GetCourse class of de-serialization :

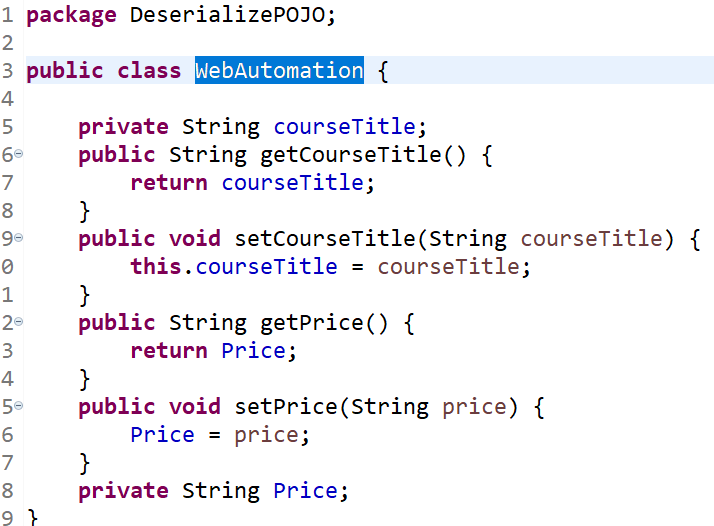




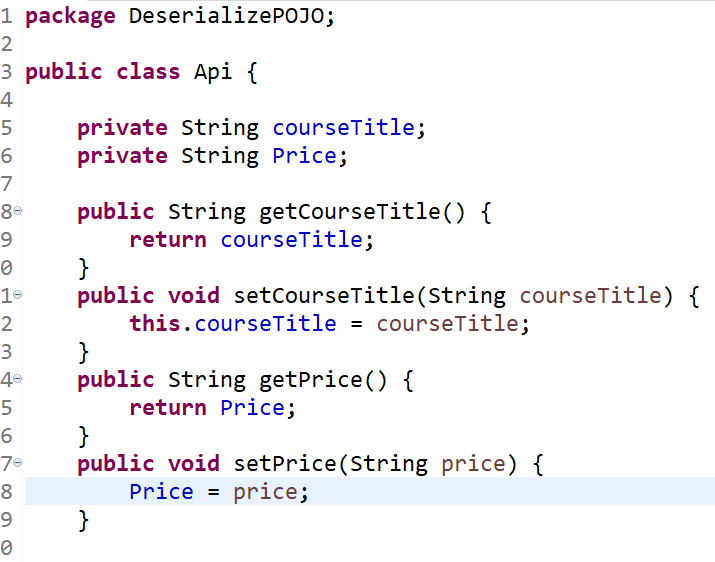
Course is an array that posses 3 sub arrays, so below class is for course



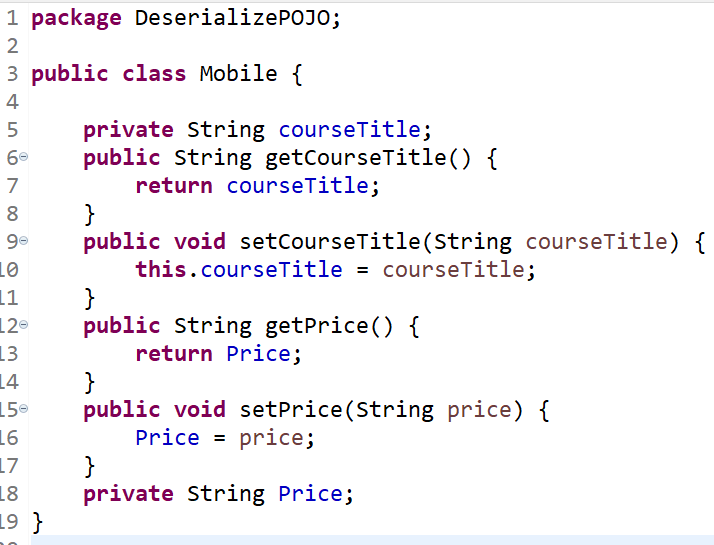
Sub array 1 : WebAutomation



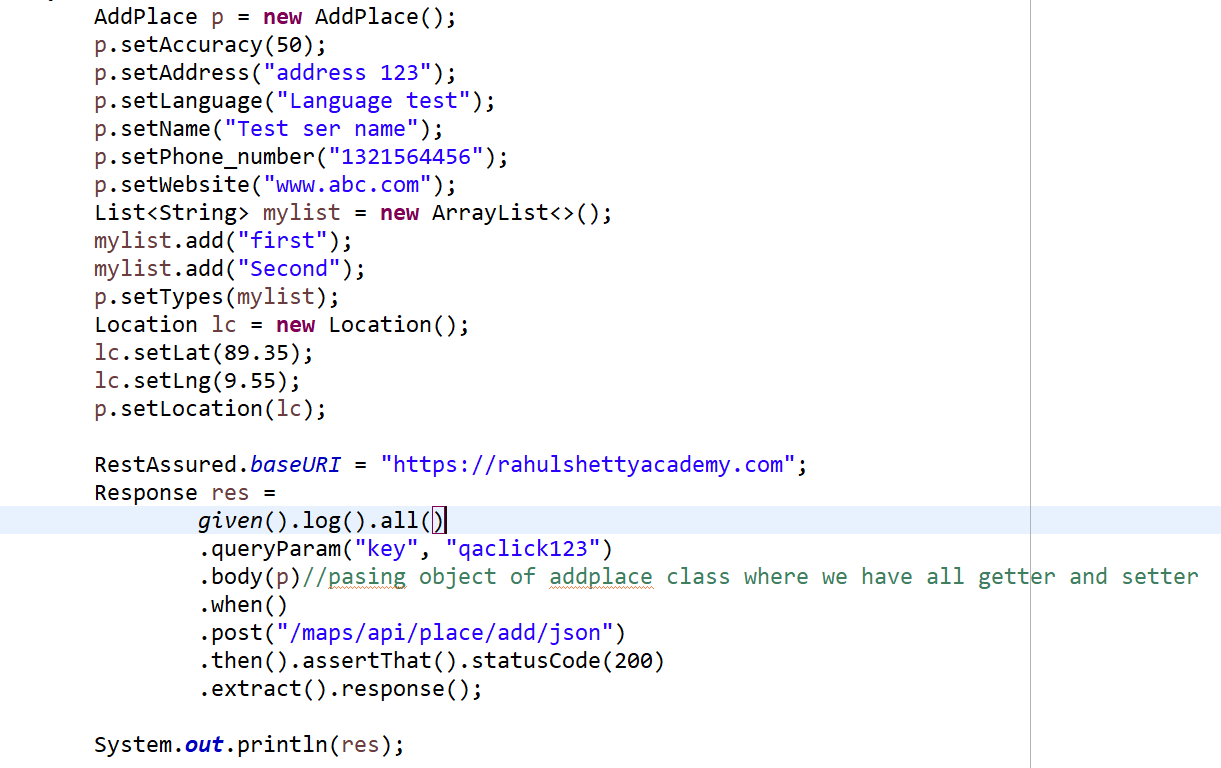
Sub array 2 : Api



Sub array 3 : Mobile



**Serialization** :



Q : What are common Json parsing techniques used in resrAssured ?

Ans : 1. Json Path 2. Deserialization using POJO class