Programming Test

Rules

- READ IT ALL FIRST (you might find it easier to do some questions in parallel)
- You can use the Internet. You need to refer to any code taken from the web (not necessary for code I have given). Not doing so is plagiarism and will be treated severely.
- No communication between students is allowed.
- You can ask questions, I decide if they are fair in an exam setting.

The What

You are required to create a web service, which provides URL end points for CRUD operations on User resources. This is to be done using Jersey. The Mortgage Calculator project can be used to form the basis of this RESTful service (as all it's XML configuration is done). The client tutorial provides a basis for a HTTP Client in Java.

The scope of this assessment does NOT include databases or persistence, as such you will NOT actually delete, or save. You will just return appropriate messages as response to requests. The web service will return ONLY JSON. The web service accepts, where appropriate, JSON input.

The Specifics

1. **Create a service to return a list of all the users (20%)**

   Create an ArrayList storing three User instances, The User class is provided on Moodle. When the request for all users is invoked, return this list to the caller in JSON format. This should be contained in an annotated method.

   Server output example:

   ```
   [ { "id": 1, "name": "Dominic", "occupation": "lecturer", "
      ↪ birthday": "Nov 1, 2016 2:02:35 PM" },
   { "id": 2, "name": "John", "occupation": "Criminal
      ↪ Mastermind", "birthday": "Nov 1, 2016 2:02:35 PM" },
   { "id": 3, "name": "Philip", "occupation": "Job Seeker", "
      ↪ birthday": "Nov 1, 2016 2:02:35 PM" } ]
   ```

2. **Create a service to return a user resource representation (20%)**

   Use the same list as before, if the user ID specified (by a URL path parameter) is present return that user in JSON format, otherwise return a sensible HTTP error code and JSON output. This should be contained in an annotated method. The user should be retrieved from the ArrayList.

   Server sample output:

   ```
   {"id":1,"name":"Dominic Carr","occupation":"Lecturer","
     ↪ birthday":"Nov 10, 2015 10:28:42 AM"}
   ```

3. **Create a service to create a user (15%)** On the server there should be a method which creates a User object from the JSON input (by automated conversion preferably); it is not necessary to save this to a database or the list. Return a sensible response, with the URL of the new resource (if this were a real system). This should all be contained in a properly annotated method.

   Example: public Response createUser(String requestBody)

   The argument requestBody will be the HTTP request body passed up from the client.

   Example input to the server:

   ```
   {"occupation": "Lecturer", "birthday": "Tue Nov 10 11:11:15
     ↪  GMT 2015", "name": "Dominic"}
   ```

   Sample server output:

   ```
   {"user_created":true,"uri":"/api/users/12"}
   ```

4. **Create a service to delete a user (20%)** Examine the input parameter and return a message indicating that the resource was removed. You will return an affirmative message stating that the resource is deleted. This should all be contained in a properly annotated method.

   Sample server output:

   ```
   {"user":{"id":1,"name":"Dominic Carr","occupation":"
     ↪ Lecturer","birthday":"Nov 10, 2015 11:08:37 AM"},"
     ↪ delete-status":"success"}
   ```

5. **Java Client (15%)**

   Create a Java client to send the requests to the service with reference to the above-defined functionality. The client should send out well-formed JSON (when necessary e.g. user creation) and display the JSON returned from the server. The client should be written using the JAX-RS client libraries as given in the tutorial files.

6. **cURL Client (10%)**

   Provide cURL commands to invoke each of the above defined HTTP requests. Examples are to be found the the relevant tutorials and example code.