Important Note:
There are two important dates for this assignment: October 14, 2016 and October 16, 2016
Midnight of October 14: Submit first version of the assignment by this date. The assignment does not have to work completely, but it has to be at the level where all startup related issues have been resolved (Eclipse, Tomcat, etc.). This submission will be *graded for 3 points*. This means if your first commit is on or before October 14th, you will receive points out of 100. If you only have one submission (after October 14th, you will receive points out of 97). The reason to have this deadline is to encourage you to get started on the assignment well ahead of the actual deadline.
Midnight of October 16: Deadline by which you should have completed your assignment.

Objective:
In this assignment you will learn about RESTEasy REST framework. It provides functionality that allows creation of JAX-RS based REST APIs.

Introduction:
Design a RESTEasy based REST API to re-model the eavesdrop website:
http://eavesdrop.openstack.org

Your task in this assignment is to create a REST API, which will invert the relationship between meetings and projects. So instead of /…/meetings/<project-name> resource model, which is used on the eavesdrop website, your REST API will model the resources as /…/projects/<project-name>/meetings.
Specifically, you will support following resources and actions on them. The exact requests and responses are colored in grey below.

1) A "projects" resource with GET action on it as follows:
 GET http://localhost:8080/assignment3/myeavesdrop/projects/
This should return all the projects listed on: http://eavesdrop.openstack.org/meetings
Response should be in XML format as follows:

```
<projects>
        <project>3rd_party_ci/</project>
         :
        <project>zun/</project>
<projects>
```

2) A "meetings" resource modeled as a sub-resource of an individual project resource. Support GET action on it as follows:
GET http://localhost:8080/assignment3/myeavesdrop/projects/{project}/meetings
E.g.:
GET http://localhost:8080/assignment3/myeavesdrop/projects/3rd_party_ci/meetings
Response should be contents from http://eavesdrop.openstack.org/meetings/3rd_party_ci/
formatted as XML format as follows:

```
<meetings>
    <year>2014/</year>
</meetings>
```

Another example request is:
GET http://localhost:8080/assignment3/myeavesdrop/projects/solum_team_meeting/meetings
Response:
```
<meetings>
    <year>2013/</year>
    <year>2014/</year>
    <year>2015/</year>
    <year>2016/</year>
</meetings>
```
You don't have to worry about maintaining a specific order for "year" XML elements in the response. The order can be whatever the order is on the eavesdrop website.
If a project does not exist on http://eavesdrop.openstack.org/meetings
your program should return "Project <project-name> does not exist". For example,
Request:
GET http://localhost:8080/assignment3/myeavesdrop/projects/non-existent-project/meetings
Response:
```
<output>
    <error>Project non-existent-project does not exist</error>
</output>
```

You should treat project names as case sensitive. So 'Solum_team_meeting' should NOT be treated as solum_team_meeting
Request:
GET http://localhost:8080/assignment3/myeavesdrop/projects/Solum_team_meeting/meetings
Response:
```
<output>
    <error>Project Solum_team_meeting does not exist</error>
</output>
```

Design Details:
You need to use RESTEasy framework for this assignment. You should organize your code into two layers, resource/controller layer and a service layer. You can divide the responsibilities between these layers as follows. Use the resource/controller layer to perform input validation and response generation. Use the service layer to implement the logic of calling the eavesdrop website and parsing the result.
Additionally, you need to implement _functional tests_ for the resource layer. Specifically, implement functional tests for following four resources:
1) GET http://localhost:8080/assignment3/myeavesdrop/projects/
2) GET http://localhost:8080/assignment3/myeavesdrop/projects/solum_team_meeting/meetings
2) GET http://localhost:8080/assignment3/myeavesdrop/projects/3rd_party_ci/meetings
3) GET http://localhost:8080/assignment3/myeavesdrop/projects/non-existent-project/meetings
A functional test tests that output of the specified <method, REST resource> combination is as expected. So in above cases, the tests need to first figure out what the correct output will be and write _asserts_ that the output is as expected. For this you will need to parse XML output that your API response generates.
For asserting the response of the GET on "projects" resource (test case 1 above), it is sufficient to assert that any 10 projects from http://eavesdrop.openstack.org/meetings exist in the response.

Submission Instructions:
Create the following folder structure:
   assignment3/src/main/java/assign/domain/<Domain class>
   assignment3/src/main/java/assign/resources/<Resource class>
   assignment3/src/main/java/assign/services/<Service classes>
   assignment3/src/test/java/assign/resources/<Resource test classes>
   assignment3/README.yaml
Use following format for README.yaml

```
---
name: SUBSTITUTE_WITH_YOUR_NAME
eid: SUBSTITUTE_WITH_YOUR_EID
notes: |
  Add some notes here.
  This can be written as block text.
```

Push the entire assignment3 folder to your bitbucket account.
Grant "read" access to Eddy and Brandon
  - Usernames: eddy_hudson, brandonhollowell
We will use the latest commit ID for grading.

Helpful Material:
a) Lecture Notes: REST API, Functional testing, XML-HTML-Parsing
b) Book Chapters: Chapters 1, 2, 3, 4, 5, 6, 17, 18, 19 of RESTful Java with JAX-RS 2.0 book.
c) Github (https://github.com/devdattakulkarni/ModernWebApps)
  Projects: REST/resteasy, Servlets/test-jsoup-1, XMLParsing
d) The ex03_1 example code from RESTful Java book
https://github.com/oreillymedia/restful_java_jax-rs_2_0

Note that for the ex03_1 example, you should change the JUnit dependency in pom.xml to use version 4.12. The JUnit test runner in Eclipse does not seem to work with the version that is there by default (4.1) in pom.xml of ex03_1 example.

Hints on getting started:
1) Read the lecture notes on REST API.
2) Download and try the resteasy example from course github page (mentioned in Helpful Material section above). Your goal should be to get this example working as the first step (either in Eclipse or Intellij or directly from command line (using mvn)).
4) Convert the resteasy example to the resource model that is required for this assignment.
5) Integrate JSoup with resteasy example.
6) Convert the domain objects from resteasy example to the domain objects required for this assignment
7) Read lecture notes on Functional testing, XML-HTML-Parsing
8) Try out the ex03_1 example from the OReilly book
9) Try out XMLParsing examples from course github page
10) Write functional tests for your assignment based on the learning from steps 8 and 9.