Getting familiar with JSON-LD
Due: 11:45 PM, October 11th, 2015

Updates:

1. Added more information for part 4.
3. Removed location-manager from list of components in Part 2.
4. Added more information to Part 3 to make it clearer.
5. Added more information to Part 4 to make the requirements clearer.
6. Requirements for Part 4 simplified.
One of the major arguments in service oriented computing is that we need deeper representations of meaning to enable interoperation among services. We’ve already studied RDF and OWL in class. This assignment aims at familiarizing you with another method of doing this.

**JSON-LD** is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases.

For the purpose of this assignment, each part will have the following types of files:

- Components
- Compositions
- Message Formats

The compositions are mapped using the components and message formats such that messages can be sent across components.

A valid composition is one which uses only message formats specified and components present such that the data sent out by one component is of the format the component receiving it can understand.

Parts 1-3 are meant to get you familiar with the concept of JSON and JSON-LD. They just use regular JSON while part 4 introduces the concept of JSON-LD.
Part 1 (10 points)
In this part you are given a composition and an incomplete component list for a virtual assistant.
The files related to this part are:
   - “part1-composition.json”
   - “part1-components.json”
   - “part1-messages.json”

Your task is to look at the files provided and define new component(s) required to complete the composition such that it is valid.

Part 2 (10 points)
You are given the following components:
   - input
   - weather-manager
   - event-manager
   - alarm-manager
   - output
Information regarding the message formats for these components have been provided in “part2-messages.json”. The components themselves are defined in “part2-components.json”.

Your task is to come up with a valid composition that uses all these components. The composition is expected to function similar to that provided in part 1.

Part 3 (30 points)
You are required to come up with two valid compositions containing your own set of components. The two compositions must have at least five components in common with each other, though you may add additional components if required.
Specify the components used, valid message formats and the compositions in three different JSON files.

For example, if you’re using components C1, C2, ..., Cn, you would have to reuse any five of these, say, C1, ..., C5 in both compositions.
Part 4 (50 points)
For this part you will be required to load the json-ld document from the compositions you created in Part 3 and print them out. This can be done using either Java or Python using some of the libraries mentioned below.

Steps to create a valid json-ld composition:
1. Attach the context as shown in the sample composition.
2. Include all of the relevant messages, components and the composition under @graph
3. Set the type fields for all of the objects to mace:message, etc.

If you wish to use Java, Jena is the framework to be used. Here is a tutorial to get a slightly older version of Jena running.
http://www.iandickinson.me.uk/articles/jena-eclipse-helloworld/

If you wish to use Python, some libraries that are useful are rdflib and pyld. Details about the libraries and instructions on how to install them are found at these URLs:

https://github.com/RDFLib/rdflib
https://github.com/RDFLib/rdflib-jsonld
https://github.com/digitalbazaar/pyld

Deliverables
1. A README.txt file containing any special instructions for your submission.
2. A zip containing:
   • The completed components and composition files from parts 1 and 2.
   • The two compositions from part 3.
   • The source code from part 4.