CPSC 501: Advanced Programming Techniques

Assignment 3: Reflective Object Serialization

Collaboration

Discussing the assignment requirements with others is a reasonable thing to do, and an excellent way to learn. However, the work you hand-in must ultimately be your work. This is essential for you to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not your original work, but is represented as such, is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code that you hand-in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:

   ```
   # the following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```

   Use the complete URL so that the marker can check the source.

2. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit code that is not primarily developed by yourself.
3. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's console, then this code is not yours.
4. Collaborative coding is strictly prohibited. Your assignment submission must be strictly your code. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modelling code after another student's algorithm. You can not use (even with citation) another student's code.
5. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
6. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

Late Penalty

Late assignments will not be accepted.

Goal

Using reflection to serialize an object to JSON and then de-serialize from JSON back to an object

Technology

Java, JSON, Git, Gitlab, Reflection

Specifics

Java 8, gitlab.cpsc.ucalgary.ca

Description

The goal of this assignment is to create two programs (that could be run on two separate computers) that communicate with each other over a network connection (see diagram below).

The **Sender** program will create an **object** (*that may have other internally referenced objects*). **Sender** will **serialize** each object created into a **JSON** String, and then send this **JSON** String as a stream of bytes to the **Receiver** program using a **socket** connection. The **Receiver** program will convert the incoming byte stream into a **JSON** String, **deserialize** the **JSON** into an **object**, and **display** the **object** to screen.

JSON is a popular dictionary style human-readable text-based storage structure. Games such as Spider-Man on PS4 and many web-based systems use **JSON** as a storage format for data. Spider-Man used it as a simple data platform to enable to creation of numerous game design tools that generated the required game world at the scale desired.

The Object Creator

This part of your system will create objects under direction of the user. The object creator must allow the user to create one or more objects from a selection of objects using some sort of text-based menu system or GUI. This menu should loop letting the user create one object, then another, and then another until they are done.

You must print out the **JSON** form of the object the user made to the shell (or your GUI) as you send it over the socket connection. This server side print out will allow you to demonstrate what is being sent.

You must be able to demonstrate that your system can handle the following kinds of objects:

1. A simple object with only primitives for instance variables. The user of your program must also be able to set the values for these fields.
2. An object that contains references to other objects. Of course, these other objects must also be created at the same time, and their primitive instance variables must be settable by the user. Your program must also be able to deal with circular references (i.e. objects connected in a graph).
3. An object that contains an array of primitives. Allow the user to set the values for the array elements to arbitrary values.
4. An object that contains an array of object references. The other objects must also be created at the same time.
5. An object that uses an instance of one of Java's collection classes to refer to several other objects. These objects, too, must be created at the same time.

The Serializer

Serialization will be implemented in a Java class called **Serializer**, and will be invoked using the method:

public static String serializeObject(Object source)

This method will serialize the complete state of the **object** passed in as a parameter and produce an **JSON** object **String** that could be stored to file, printed out, or sent over a network connection. You can use a library to manage the creation of **JSON** objects, but must perform the reflection process yourself to put data in these objects.

The base of the **JSON** container must be a list of **objects**. The base object being sent will be first, and then any objects it references will be stored after it within this list. For example:

{"objects":[

{"class":"class_name",…},

{"class":"class_name",…}

]}

An entry for an object will have 4 entries. The name of the **class**, the unique integer **id** of the object (most likely created using **IdentityHashMap**), the **type** of the object (***object***/***array*** are the two types of types), and an entry which will contain each field of in **fields**. For example:

{"class":"Solution.ObjectA","id":"0","type":"object","fields":[…]}

Fields will be a list of the fields for the object. These fields may have been declared in the object itself, or by its parent, or parent's parent, etc. For example:

“fields":[

{"name":"x",…},

{"name":"y",…}

]

Each **field** will contain the field’s **name**, the field’s **declaring class**, and the ***value*/*reference*** of the field.

If the type of the field is a primitive, then store the **value** of the field.

{name":"field_name","declaringclass":"class_name","value":"1.0"}

If the field is a reference to a null.

{name":"field_name","declaringclass":"class_name","reference":"null"}

If the field is a reference to another object, store the ***IdentityHashMap*** **id** of that object as content of a reference element. For example:

{name”:”field_name”,”declaringclass”:”class_name”,”reference”:”1”}

Any additional object referenced by the root object, such as by the above **reference id** will also need to be serialized and stored in the root list of **objects** with the proper reference **id**. Each unique object should only be stored once, even if it is reference multiple times.

Array objects will be similar to the object element described first. We will use the type **array** to indicate that this is an array. We will also add the additional **length** attribute is used, and each element of the array will be stored in the list. As with fields a value entry will be stored as **value** and an object as a **reference.** For example:

{"class":"[I","id":"1","type":"array","length":"5","entries":[

{"value":"0"},

{"value":"0"},

{"value":"0"},

{"value":"3"},

{"value":"0"}

]}

The Network Connection

Java provides the **java.net.Socket** and **java.net.ServerSocket** class to help implement a network connection between two programs. Initially, while testing your system, you can run the two programs on the same computer. When demonstrating your system to the TAs must have evidence that your system would work between two computers. It must be clear that the code is working through a socket connection with **JSON** serialization.

The Deserializer

Deserialization will be implemented in a Java class called **Deserializer**, and will be invoked using the method:

public static Object deserializeObject(String source)

This method will deserialize a **JSON** object string passed in as parameter, returning the reconstituted object (and any objects it refers to).

The Visualizer

This part of your system displays the deserialized objects to screen. You can use a text-based or graphical user interface for this. The object **Inspector** you created in **Assignment 2** may be modified to be used here in a text-based system. (Unlike Assignment 2 you only have to print class identifying information and field values and only have to fully follow recursion for each unique object once.) Be sure that this part of the system shows that the deserialized objects are properly recreated.

MANDATORY Requirement

You must demonstrate a working program to your TA after the due date. **This is mandatory and you will not receive a grade until you do so.** Make certain you show that all requirements of the system have been fulfilled and implemented correctly. You must also use version control, unit testing, and refactoring throughout this assignment as in prior assignments. The demonstration includes being able to reference your source code to explain how it works.

Submit the following using the Assignment 3 Dropbox in D2L:

1. The complete source of your source code and your unit testing code.
2. Directions/access to the **gitlab.cpsc.ucalgary.ca** project so that your TA can access your project and read your report.

Bonus:

Create a variant of your program that works with **XML as well as JSON**. Your objects should be serializable to **either JSON or XML** by the **Sender** and the **Receiver** should be able to deserialize from **both JSON and XML** depending on what is received. The bonus must be demonstrated to the TA for grading as well.

Grading

Object creation	Simple, references, array primitives, array references, java.util.Collection	10	_____
JSON serialize	Simple, references, array primitives, array references, java.util.Collection	10	_____
JSON deserialize	Simple, references, array primitives, array references, java.util.Collection	10	_____
Object visualize		4	_____
Server/Client		4	_____
Version Control		4	_____
Tests/Refactoring		4	_____
Design quality		4	_____
Total		**50**	_____
Bonus	JSON and XML format	5	_____

Letter	**A+**	**A**	**A-**	**B+**	**B**	**B-**	**C+**	**C**	**C-**	**D+**	**D**	**F**
Points	**47.5**	**45**	**42.5**	**40**	**37.5**	**35**	**32.5**	**30**	**27.5**	**25**	**22.5**	**<22.5**