CPSC 501 – Assignment 3 – Distributed Objects using Reflective Serialization/Deserialization Refactorings

In this refactoring document, I'm using the same format of questions as asked in Assignment 1.

The version controlled code is hosted in a public repository on GitHub, which can be found here: https://github.com/natejacko/CPSC501-Assignment3

The below refactoring can be found with changes associated to the GitHub hash code: fdfc15

What needed to be improved?
After implementing the serializer and starting to implement the deserializer, the code began to grow to a single long method (code smell) `Object deserialize(Document doc)`. This also led to numerous sections of duplicate code, especially when checking types for both fields and arrays. A change was needed to remove the duplicate code, make the method more readable, and allow for an easier time to debug future issues that might've arisen whilst testing.

What refactoring was applied?
The main refactoring applied was to extract methods. Extracting methods was done to shorten the one long deserialize method, into logical groupings of functionality, and to slim down the duplicated code when checking types for fields and arrays.

What code in which files was altered?
Deserializer.java was the only file altered in this refactoring. The method `Object deserialize(Document doc)` had functionality extracted and the following methods were added:

- `private void deserializeInstances(HashMap<String, Object> instances, List<Element> elements)`
- `private void deserializeFields(HashMap<String, Object> instances, List<Element> elements)`
- `private Object getValueFromElement(Element e, Object obj, Class c, HashMap<String, Object> instances)`

  The last method added (getValueFromElement) was the extracted method to reduce the duplicate code and was then able to be called by both field and array setters.

How was the code tested?
The code was tested further on by some JUnit tests, and by cross-referencing similar deserialization code available in the Java Reflection in Action textbook.

Why is the code better structured after this refactor?
The code is better structured as it can accommodate for future development in the deserialization class if needed. Additionally, it covers the two code smells outlined above (long method – deserialize method was drastically decreased in length. duplicate code – type checking was reduced). Finally, the code became more readable and method names gave a lot more meaning after the refactor