Benton Luu 10175103

CPSC 501 – Assignment 3: Serialization/Deserialization

Link to GitHub Repository: https://github.com/bentonluu/CPSC501-Assignment3-Serialization-Deserialization

Refactoring 1: Long Method -> Extract Method
(GitHub SHA: 9ca0c06928d038204a61497968433b4f2403738d)

- In the method 'deserialize', all of the responsibility to create the object/array instance and set the values for the created instance was done within that singular method. Thus, since all of the functional code was done in that one method it made the method long and complicated.
- Using the refactoring method of Extract Method, extracted 2 methods from the 'deserialize' method; 'setObjFieldValue' and 'initializeObject'. The 'setObjFieldValue' method was used to set the values of the created object/array instance from what was provided from the XML document sent over. The 'intializeObject' method was used to create the object/array instance.
- To test that the refactoring was applied correctly, created 6 JUnit tests ('deserializeClassA, 'deserializeValueClassA, deserializeClassB, 'deserializeClassC, 'deserializeClassD, and 'deserializeClassE) to verify that the deserialized object/array returned from the 'deserialize' method were correctly re-created.
- The improvement from this refactoring was that it reduced the complexity of the method 'deserialize', by splitting the responsibility into two distinct methods. By separating the responsibility into their own methods it overall made the code more understandable in what each method does and troubleshoot issues in the future.

BEFORE

Deserializer.java

```
public class Deserializer {
   private Object deserialize(Document document) throws Exception {
      …
      Object objInstance = null;
      Element rootElement = document.getRootElement();
      List<Element> objectList = rootElement.getChildren();
      for (Element object : objectList) {
          …
      }
      return objInstance
   }
   …
}
```

AFTER

Deserializer.java

```
public class Deserializer {
   private Object deserialize(Document document) throws Exception {
      …
      setObjFieldValue(objectList);
      return ihm.get("0");
      }

   public void setObjFieldValue(List<Element> objectList) throws Exception {
      for (Element object : objectList) {
         Object objInstance = initializeObject(object);
         …
      }
   }

   public Object initializeObject(Element object) throws Exception {
      Object objInstance;
      Class classObj = Class.forName(object.getAttributeValue("class"));
      …
      ihm.put(object.getAttributeValue("id"), objInstance);
      return objInstance;
   }
   …
}
```

Refactoring 2: Long parameter list -> Preserve Whole Object
(GitHub SHA: 563eda681627c16e1d32fc94b662a37044f759ca)

- The parameter list to obtain the value within an Element instance was quite long with nested method calls needed in order to get the required value.
- Using the refactoring method Preserve Whole Object, replaced the 'objFieldList.get(i)' method call with a local Element variable named 'objElement' instead in order to preserve the whole object and updated all associated parameter lists with the new variable.
- Re-used the previous JUnit tests created for the deserializer to ensure that object/array instances were properly re-created after the implement the refactoring method.
- The enhancement to the code from the refactoring method being used was that it shortened the length of the parameter lists in order to obtain a value from an Element object. It made it easier to understand the actual method was doing as well as simplified the process of resolving an issue since the Element object is now associated with the common local variable.

BEFORE

Deserializer.java

```
public class Deserializer {
  private Object deserialize(Document document) throws Exception {
    …
    setObjFieldValue(objectList);
    return ihm.get("0");
    }

  public void setObjFieldValue(List<Element> objectList) throws Exception {
    for (Element object : objectList) {
    …
          if (objInstance.getClass().isArray()) {
            if (objFieldList.get(i).getName().equals("value")) {
              Array.set(objInstance, i,
          parseFieldValue(objInstance.getClass().getComponentType(),
          objFieldList.get(i));
              }
            else if (objFieldList.get(i).getName().equals("reference")) {
              Array.set(objInstance, i, ihm.get(objFieldList.get(i).getText()));
              }
            else if (objFieldList.get(i).getText().equals("null")) {
              Array.set(objInstance, i, null);
              }
          }
        …
    }
  }
…
}
```

AFTER

Deserializer.java

```
public class Deserializer {
  private Object deserialize(Document document) throws Exception {
    …
    setObjFieldValue(objectList);
    return ihm.get("0");
    }

  public void setObjFieldValue(List<Element> objectList) throws Exception {
    for (Element object : objectList) {
    …
          Element objElement = objFieldList.get(i);
          if (objInstance.getClass().isArray()) {
            if (objElement.getName().equals("value")) {
              Array.set(objInstance, i,
          parseFieldValue(objInstance.getClass().getComponentType(), objElement));
              }
            else if (objElement.getName().equals("reference")) {
              Array.set(objInstance, i, ihm.get(objElement.getText()));
              }
            else if (objElement.getText().equals("null")) {
              Array.set(objInstance, i, null);
              }
          }
        …
    }
  }
…
}
```