OBJECT ORIENTED ANALYSIS AND DESIGN WITH JAVA
LAB 6

Submitted by: Mahika Gupta
SRN: PES1UG20CS243
Date: 10/03/2023

Java Serialization and Deserialization with Hashmap

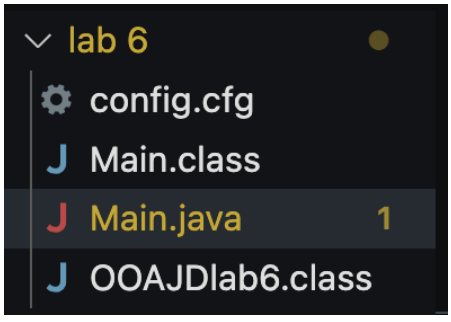
1. Before config.cfg file exists: on running program for the first time:

```
(base) mahika@the-book lab 6 % cd "/Users/mahika/OOADJ LAB/lab 6/" && javac Main.java && java Main


Key = PATH
Value = /Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/miniforge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Applications/VMware Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin


Key = Version
Value = 13.1


Key = SystemName
Value = macOS Ventura
(base) mahika@the-book lab 6 %
```



We can see that config.cfg file is created and added to directory and its contents are printed.

2. After config.cfg file exists: on running program for the second time:

```
(base) mahika@the-book lab 6 % cd "/Users/mahika/OOADJ LAB/lab 6/" && javac Main.java && java Main
Before adding Model:


Key = PATH
Value = /Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/miniforge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Applications/VMware Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin


Key = Version
Value = 13.1


Key = SystemName
Value = macOS Ventura
After adding Model:


Key = PATH
Value = /Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/miniforge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Applications/VMware Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin


Key = Version
Value = 13.1


Key = SystemName
Value = macOS Ventura


Key = MODEL
Value = M1 MacBook Air
(base) mahika@the-book lab 6 %
```

We can see that the contents before updating the file are printed first, and then printed after updating, with the new MODEL key value pair.

3. After config.cfg file exists and changes are made: on running program for the third time:

```
(base) mahika@the-book lab 6 % cd "/Users/mahika/OOADJ LAB/lab 6/" && javac Main.java && java Main
Before adding Model:


Key = PATH
Value = /Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/miniforge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Applications/VMware Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin


Key = Version
Value = 13.1


Key = SystemName
Value = macOS Ventura


Key = MODEL
Value = M1 MacBook Air
After adding Model:


Key = PATH
Value = /Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/miniforge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Applications/VMware Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin


Key = Version
Value = 13.1


Key = SystemName
Value = macOS Ventura


Key = MODEL
Value = M1 MacBook Air
(base) mahika@the-book lab 6 %
```

We can see that since config.cfg has been updated already, the updated contents are printed both times.

Serialization:

Serialization refers to converting the state of an object into a byte stream. This byte stream includes all of the objects data, including its instance variables and references to other objects. The object's class information is also included in the byte stream which allows the object to be reconstructed later. Java provides a built-in mechanism for serialization through the Serializable interface. To make an object serializable, the class must implement this interface, which is a marker interface with no methods. Additionally, any fields that should not be serialized must be marked as transient.

- **The child class doesn't have to implement the Serializable interface, if the parent class does**
- **The serialization process only saves non-static data members, but not static or transient data members**

Deserialization:

Deserialization is the process of reconstructing an object from a byte stream. When an object is deserialized, its instance variables are set to the values stored in the byte stream, and any references to other objects are restored. However, you must have the definition of the object to successfully recreate it.

Serialization and Deserialization allow Java objects to be stored or transmitted in a persistent form.

Hashmap:

HashMap is a data structure that allows efficient mapping of key-value pairs. Provides functionality of Hash Table. Here, keys are unique identifiers used to associate each value on a map. Key-Value pairs can be inserted by the put method. Key-Value pairs can be fetched by the get method. The main advantage of using HashMap Object is that it provides O(1) access time. It is implemented using an Array and a Linked List.

Program:

```
import java.util.*;
import java.io.*;


class OOAJDlab6 implements Serializable{
   HashMap<String,String> map = new HashMap<String,String>();

   OOAJDlab6(){

this.map.put("PATH","/Users/mahika/.nvm/versions/node/v16.14.2/bin:/Users/mahika/minif
orge3/bin:/Users/mahika/miniforge3/condabin:/opt/homebrew/bin:/opt/homebrew/sbin:/opt/
local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/opt/h
omebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:usr/local/
mysql-8.0.30-macos12-arm64/bin:/usr/local/mysql-8.0.30-macos12-arm64/support-files:/Ap
plications/VMware
Fusion.app/Contents/Public:/Library/Apple/usr/bin:/Users/mahika/.local/bin");
       this.map.put("Version","13.1");
       this.map.put("SystemName","macOS Ventura");
   }

   void displayhashmap(){
       for(Map.Entry<String,String> entry: map.entrySet()){
           System.out.println("\n\nKey = " + entry.getKey() + " \nValue = " +
entry.getValue());
       }
   }

   void insertKey(String key, String value){
       this.map.put(key,value);
   }

   void remove(String key){
       this.map.remove(key);
   }

}

class Main{
   public static void main(String args[]){
       File dir = new File("./");
```

```
        File dirList[] = dir.listFiles();
        File f = new File("./config.cfg");

        if(f.exists()){
            try{
                ObjectInputStream in = new ObjectInputStream(new
FileInputStream("./config.cfg"));
                OOAJDlab6 conf = (OOAJDlab6)in.readObject();
                in.close();
                System.out.println("Before adding Model: ");
                conf.displayhashmap();
                conf.insertKey("MODEL", "M1 MacBook Air");
                System.out.println("After adding Model: ");

                conf.displayhashmap();
                FileOutputStream fout = new FileOutputStream("./config.cfg");
                ObjectOutputStream out = new ObjectOutputStream(fout);
                out.writeObject(conf);
                out.flush();
                out.close();
            }

            catch(Exception e){
                System.out.println(e);
            }
        }

        else{
            try
             {
                 OOAJDlab6 conf = new OOAJDlab6();
                 conf.displayhashmap();
                 FileOutputStream fout = new FileOutputStream("./config.cfg");
                 ObjectOutputStream out = new ObjectOutputStream(fout);
                 out.writeObject(conf);
                 out.flush();
                 out.close();
             }
             catch(Exception e)
             {
                 System.out.println(e);
             }
```

```
        }
    }
}
```