**Object Oriented Analysis and Design using Java**

**Self Learning: Java Serialization, Deserialization and HashMap**

|  |  |  |  |
| --- | --- | --- | --- |
| Vanshika Goel | PES1UG20CS484 | Section H | Roll No 40 |

**Serialization** is a mechanism of converting the state of an object into a byte stream.

It is mainly used to travel object's state on the network (that is known as marshalling).

Serializable is a marker interface (has no data member and method). It is used to "mark" Java classes so that the objects of these classes may get a certain capability.

**Deserialization** is the reverse process where the byte stream is used to recreate the actual Java object in memory.

The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

**HashMap<K, V>** stores the data in (Key, Value) pairs, and the data can be accessed by an index of another type (e.g. an Integer).

One object is used as a key (index) to another object (value). If you try to insert the duplicate key, it will replace the element of the corresponding key.

**HashMap** allows to store the null keys as well, but there should be only one null key object and there can be any number of null values.  This class makes no guarantees as to the order of the map. To use this class and its methods,  **java.util.HashMap** package or its superclass needs to be imported.

**Code:**

import java.io.\*;

import java.util.\*;

public class ConfigJava implements Serializable

{

private HashMap<String, String> configJava;

public ConfigJava()

{

configJava = new HashMap<String, String>();

}

public void setVal(String key, String value)

{

configJava.put(key, value);

}

public String getVal(String key)

{

return configJava.get(key);

}

public static void main(String[] args)

{

ConfigJava configJava;

File fileconfig = new File("/Users/vanshikagoel/Desktop/OOAD/Lab/Week 7/config.cfg");

if (fileconfig.exists())

{

try

{

FileInputStream fileInputStream = new FileInputStream(fileconfig);

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

configJava = (ConfigJava) objectInputStream.readObject();

objectInputStream.close();

}

catch (IOException | ClassNotFoundException e)

{

e.printStackTrace();

return;

}

}

else

{

configJava = new ConfigJava();

}

configJava.setVal("Path", "//Users/vanshikagoel/PES1UG20CS484/");

configJava.setVal("Version", "12.3.1");

configJava.setVal("SystemName", "PES1UG20CS484");

configJava.setVal("Language", "English");

configJava.setVal("Operating System", "macOS");

try

{

FileOutputStream fileOutputStream = new FileOutputStream(fileconfig);

ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);

objectOutputStream.writeObject(configJava);

objectOutputStream.close();

}

catch (IOException e)

{

e.printStackTrace();

return;

}

System.out.println("config.cfg file has these values:");

System.out.println("1. Path: " + configJava.getVal("Path"));

System.out.println("2. Version: " + configJava.getVal("Version"));

System.out.println("3. System\_Name: " + configJava.getVal("SystemName"));

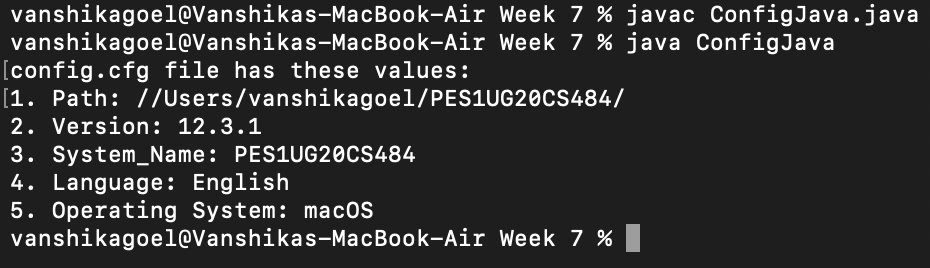
System.out.println("4. Language: " + configJava.getVal("Language"));

System.out.println("5. Operating System: " + configJava.getVal("Operating System"));

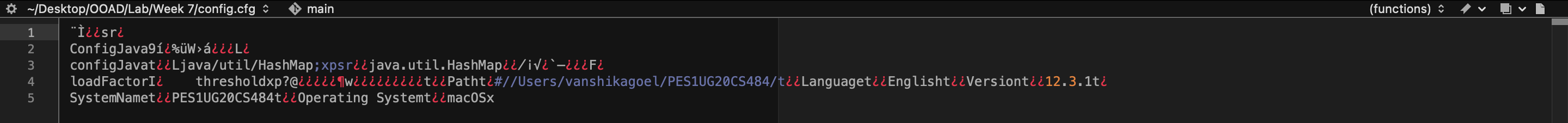
}

}

**Output Screenshot:**

****

**Config.cfg file:**

****