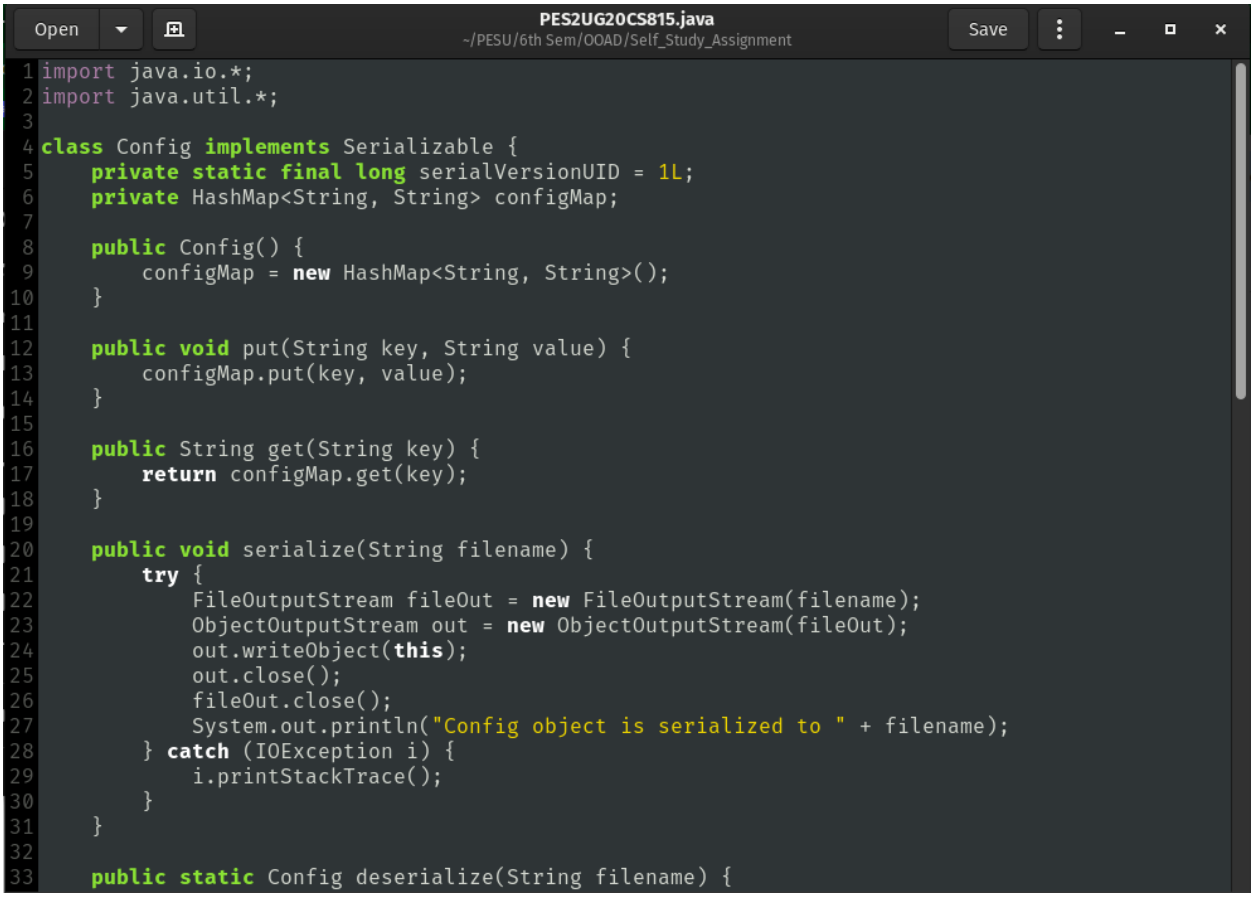
UE20CS352-OOAD SELF-LEARN-ASSIGNMENT

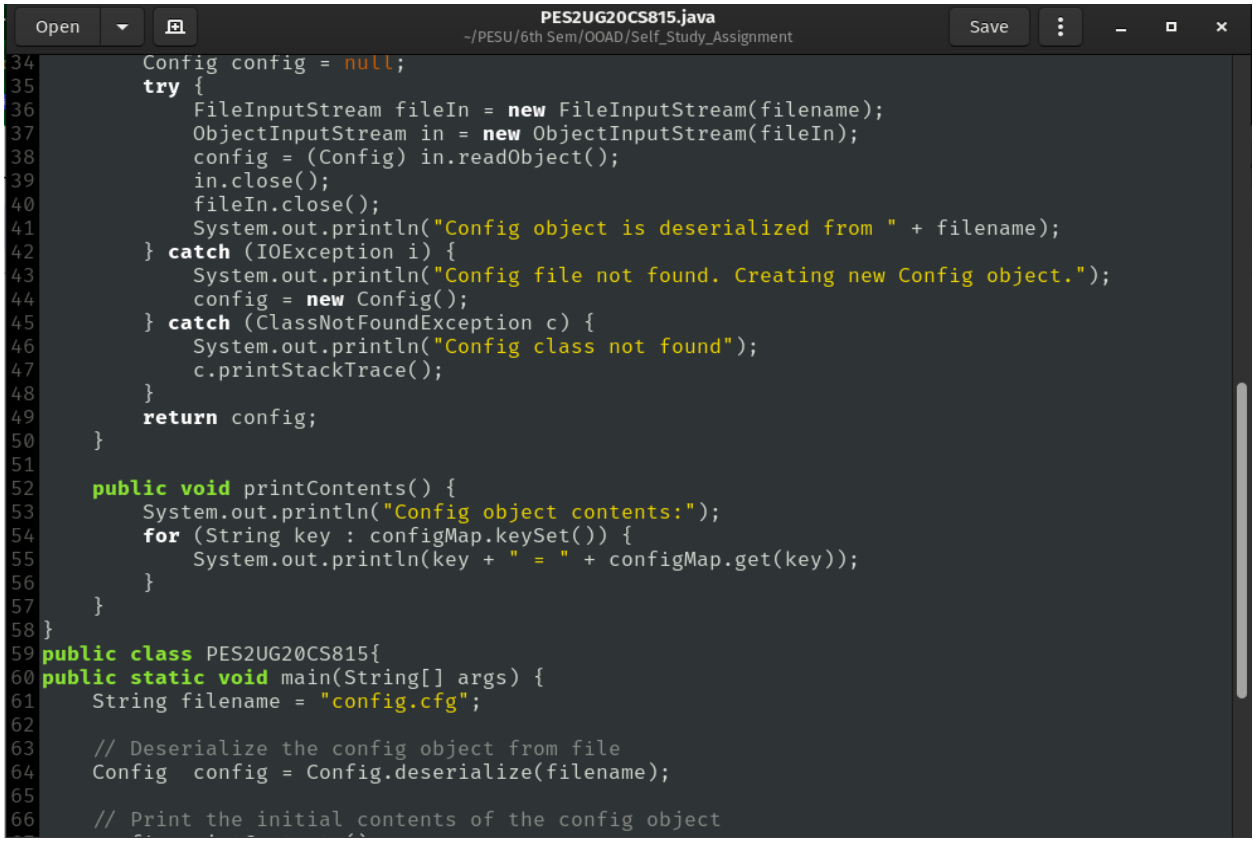
NAME	SRN	CLASS & SECTION
VIJAY J	PES2UG20CS815	6 - J

Java Serialization and Deserialization. Write a program using these concepts with the use of HashMap data structure.

Code:

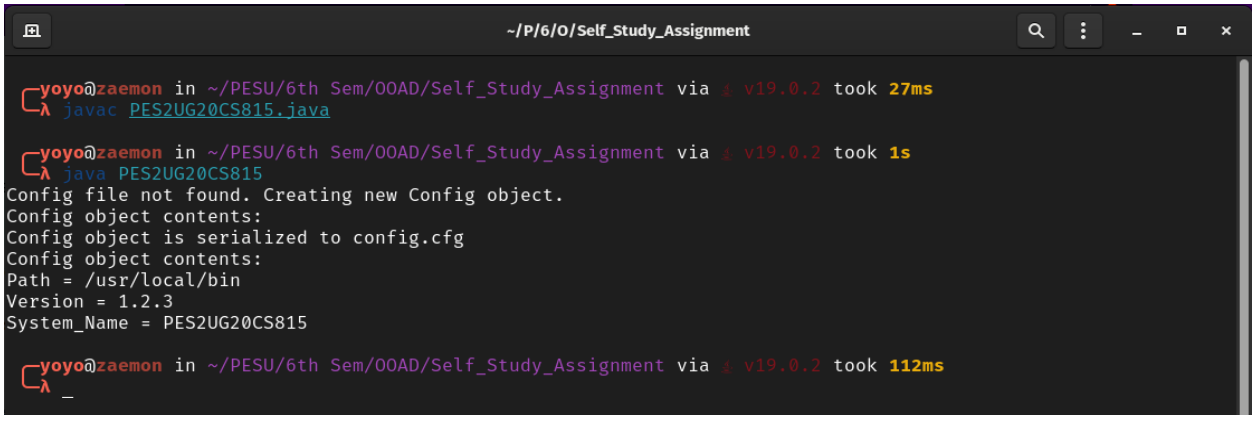



```java
import java.io.*;
import java.util.*;

class Config implements Serializable {
    private static final long serialVersionUID = 1L;
    private HashMap<String, String> configMap;

    public Config() {
        configMap = new HashMap<String, String>();
    }

    public void put(String key, String value) {
        configMap.put(key, value);
    }

    public String get(String key) {
        return configMap.get(key);
    }

    public void serialize(String filename) {
        try {
            FileOutputStream fileOut = new FileOutputStream(filename);
            ObjectOutputStream out = new ObjectOutputStream(fileOut);
            out.writeObject(this);
            out.close();
            fileOut.close();
            System.out.println("Config object is serialized to " + filename);
        } catch (IOException i) {
            i.printStackTrace();
        }
    }

    public static Config deserialize(String filename) {
```

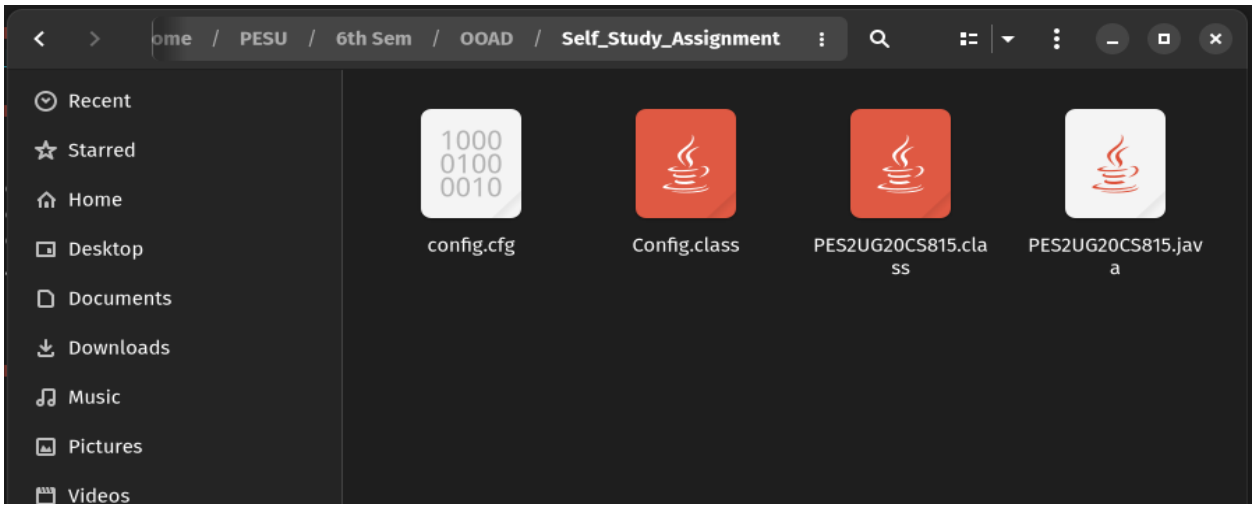



```java
        Config config = null;
        try {
            FileInputStream fileIn = new FileInputStream(filename);
            ObjectInputStream in = new ObjectInputStream(fileIn);
            config = (Config) in.readObject();
            in.close();
            fileIn.close();
            System.out.println("Config object is deserialized from " + filename);
        } catch (IOException i) {
            System.out.println("Config file not found. Creating new Config object.");
            config = new Config();
        } catch (ClassNotFoundException c) {
            System.out.println("Config class not found");
            c.printStackTrace();
        }
        return config;
    }

    public void printContents() {
        System.out.println("Config object contents:");
        for (String key : configMap.keySet()) {
            System.out.println(key + " = " + configMap.get(key));
        }
    }
}
public class PES2UG20CS815{
public static void main(String[] args) {
    String filename = "config.cfg";

    // Deserialize the config object from file
    Config  config = Config.deserialize(filename);

    // Print the initial contents of the config object
    config.printContents();

    // Update the config object
    config.put("Path", "/usr/local/bin");
    config.put("Version", "1.2.3");
    config.put("System_Name", "PES2UG20CS815");

    // Serialize the updated config object to file
    config.serialize(filename);

    // Print the updated contents of the config object
    config.printContents();
}
}
```

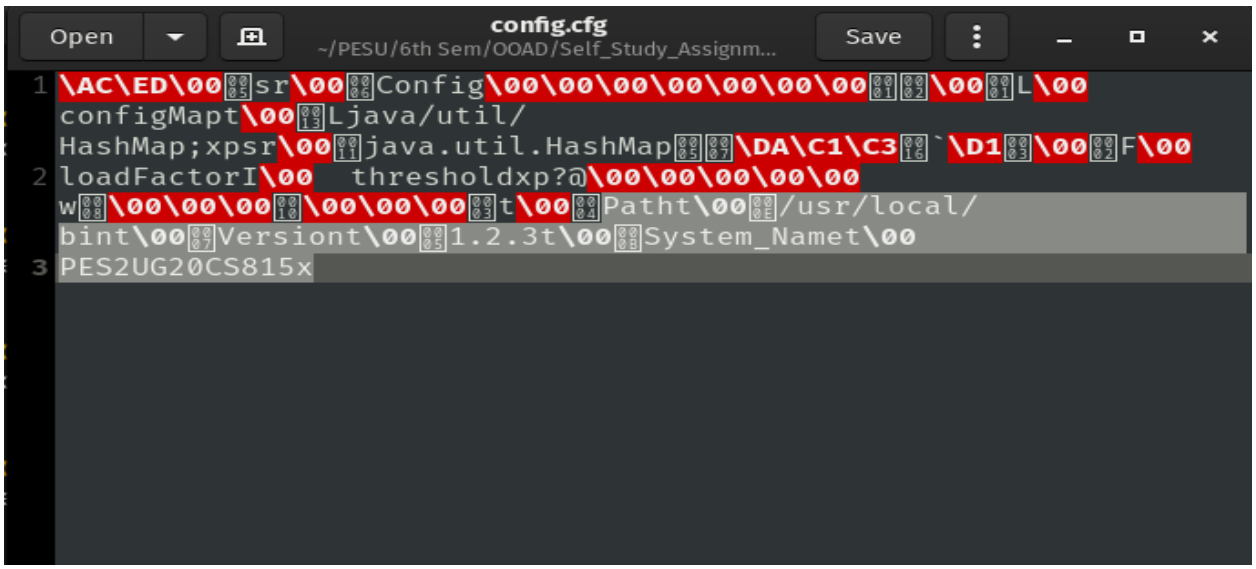
Output:



config.cfg created



config.cfg file



Summary:

Serialization:

Serialization in Java is the process of converting an object's state into a sequence of bytes so that it can be easily stored and transmitted across different applications and systems. This serialized data can then be deserialized later to recreate the original object.

To serialize an object in Java, the class needs to implement the Serializable interface, which is a marker interface that indicates that the class can be serialized. The Serializable interface has no methods, but it serves as a signal to the Java runtime that the object can be serialized.

Once the class has implemented the Serializable interface, the serialization process can be performed by creating an instance of ObjectOutputStream and passing it a stream object, which can be a FileOutputStream or any other type of output stream. The writeObject() method of ObjectOutputStream is used to write the object to the stream, which will serialize the object's state and write it to the stream.

Deserialization:

In Java, serialization is the process of converting an object into a stream of bytes so that it can be stored in a file or transmitted over a network. Deserialization is the process of converting that stream of bytes back into an object.

Deserialization in Java is done using the ObjectInputStream class, which reads data from an InputStream and reconstructs the serialized object. The ObjectInputStream reads the stream of bytes and creates an object of the serialized class.

HashMap:

Java **HashMap** class implements the Map interface which allows us to store key and value pair, where keys should be unique. If you try to insert the duplicate key, it will replace the element of the corresponding key. It is easy to perform operations using the key index like updation, deletion, etc. HashMap class is found in the java.util package.

HashMap in Java is like the legacy Hashtable class, but it is not synchronized. It allows us to store the null elements as well, but there should be only one null key. Since Java 5, it is denoted as HashMap<K,V>, where K stands for key and V for value. It inherits the AbstractMap class and implements the Map interface.