OOAD

Lab 6 **-Adithya M, Section K**
-PES1UG20CS621

Serialization:

Serialization is the process of converting an object into a byte stream, which can be used to store the state of an object or transmit it over a network. In Java, serialization is implemented using the Serializable interface. This interface does not have any methods, but it is used to indicate that a class can be serialized. When an object of a class that implements Serializable is serialized, its fields are converted into a byte stream.

Deserialization:

Deserialization is the reverse process of serialization, where the byte stream is converted back into an object. In Java, deserialization is implemented using the ObjectInputStream class. This class is used to read a byte stream and create an object from it.

HashMap:

HashMap is a data structure in Java that is used to store key-value pairs. It is a part of the java.util package and implements the Map interface. HashMap stores its elements in a hash table, which allows for fast retrieval of elements based on their keys. HashMap does not allow duplicate keys, and each key can map to at most one value.

Config.java

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

public class Config implements Serializable {
   private static final long serialVersionUID = 1L;
   private HashMap<String, String> map = new HashMap<>();

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

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

   public void serialize() {
      try {
         FileOutputStream fileOut = new FileOutputStream("config.cfg");
         ObjectOutputStream out = new ObjectOutputStream(fileOut);
         out.writeObject(this);
         out.close();
```

```
            fileOut.close();
        } catch (IOException i) {
            i.printStackTrace();
        }
    }

    public static Config deserialize() {
        Config config = null;
        try {
            FileInputStream fileIn = new FileInputStream("config.cfg");
            ObjectInputStream in = new ObjectInputStream(fileIn);
            config = (Config) in.readObject();
            in.close();
            fileIn.close();
        } catch (IOException i) {
            i.printStackTrace();
        } catch (ClassNotFoundException c) {
            System.out.println("Config class not found");
            c.printStackTrace();
        }
        return config;
    }
}
```

Application.java

```
import java.io.*;

public class Application {
    public static void main(String[] args) {
        Config config = null;
        String filename = "config.cfg";

        File file = new File(filename);
        if (file.exists()) {
            try {
                FileInputStream fis = new FileInputStream(filename);
                ObjectInputStream ois = new ObjectInputStream(fis);
                config = (Config) ois.readObject();
                ois.close();
                fis.close();
            } catch (IOException | ClassNotFoundException e) {
                e.printStackTrace();
            }
        } else {
            config = new Config();
```

```
        }

        config.put("Path", "C:\\Program Files\\Java\\jdk1.8.0_121\\bin");
        config.put("Version", "4.2.0");
        config.put("System_Name", "Adi-PC");

        config.serialize();

        System.out.println("Updated Config Values:");
        System.out.println("Path: " + config.get("Path"));
        System.out.println("Version: " + config.get("Version"));
        System.out.println("System_Name: " + config.get("System_Name"));
    }
}
```

Output: