Java Serialization

Introduction

- Serialization is simply turning an existing object into a byte array.
- This byte array represents
  - the class of the object,
  - the version of the object,
  - and the internal state of the object.
- This byte array can be used to
  - Store the state of the object for later retrieval (store in a file).
  - Send object over a network to another JVM running the same code.
- Deserialization is converting a byte array back to an Object.

Serialization

- A marker interface called Serializable is provided.
  - It is an empty interface, hence, classes implementing this interface are not required to override any methods related to the interface.
- ObjectOutputStream.writeObject(Object) traverses all the internal references of the object recursively and writes all of them provided classes associated with the objects have also implemented Serializable.
- Any fields that the programmer does not want to serialize should be marked as transient.

Object Storage in a File

- Classes **ObjectInputStream** and **ObjectOutputStream** are high-level streams that contain the methods for serializing and deserializing an object.
- Writing Objects to a File
  1. FileOutputStream file = new FileOutputStream("Student1.ser");
  2. The writeObject(Object obj) from the ObjectOutputStream class can be used to write an object into a file
- Reading Objects from a File

Classes Implementing the Serializable Interface

```
public class Student implements Serializable {
        private int Roll;
        private String Name;
        private LibraryAccount lib;
        private final static long serialVersionUID =  1000;
        public Student(int Roll, String Name){
                this.Roll= Roll;
                this.Name = Name;
        }
        public int getRoll(){return this.Roll;}
        public String getName(){return this.Name;}
        public LibraryAccount getLibAccount(){
                return this.lib;}
        public void createLibraryAccount(){
        this.lib = new LibraryAccount();}
}
```

```
public class LibraryAccount implements Serializable {
        private int Id;
        private static int NumberAccounts=0;
        public LibraryAccount(){
                NumberAccounts++;
                this.Id= NumberAccounts;
        }
        public int getId(){return this.Id;}
        public static int getNumberofAccounts() {
                return NumberAccounts;}
}
```

Writing Objects in a File

```
public class WriterClass {
        public static void main(String[] args) {
                Student std = new Student(1,"Muhammad Ali");
                std.createLibraryAccount();
                System.out.println(std.getRoll()+ " " + std.getName() + " " +
std.getLibAccount().getId());
                try {   /* A file is created/opened where we want to write an object. “ser” is normally a
                        convention that is followed for file names containing a serialized file.*/
                        FileOutputStream file = new FileOutputStream("Student1.ser");
                        / *The writeObject(Object obj) from the ObjectOutputStream class can be used to
                        write an object into a file.*/
                        ObjectOutputStream out = new ObjectOutputStream(file);
                        out.writeObject(std);
                        out.close();
                        file.close();}
                catch (FileNotFoundException e) {e.printStackTrace();}
                catch (IOException e) {e.printStackTrace();}
} }
```

Reading Object (Deserialization) from a File

```
public class ReaderClass {
        public static void main(String[] args) {
                try {       /* The file is opened from where we want to read an object.*/
                        FileInputStream input = new FileInputStream("Student1.ser");
                        / *The readObject() from the ObjectInputStream class can be used to
                        read an object from a file. The return object needs to be casted before use.*/
                        ObjectInputStream inStream = new ObjectInputStream(input);
                        Student obj = (Student) inStream.readObject();
                        System.out.println(obj.roll+ " " + obj.name);
                        inStream.close();
                        input.close();
                        }
                catch (FileNotFoundException e) {e.printStackTrace();}
                catch (IOException e) {e.printStackTrace();          }
                catch (ClassNotFoundException e) {e.printStackTrace();}
        }
}
```

Sending Objects over a Network

- Example attached separately in the shared folder.