How to serialize and de-serialize ArrayList in Java

December 13, 2016 SJ Serialization 0

In this article, we will discuss how to serialize list of objects and also deserializing the same.

Already we have seen how to serialize and de-serialize objects in Java i.e.;

- Serialization and De-Serialization in Java
- Serialization with Aggregation (HAS-A relationship)
- Serialization with Inheritance (IS-A relationship)
- Importance of SerialVersionUID in Serialization

When we discussed above topics, we concentrated only on single Object i.e.; POJO (Plain Old Java Object)

Here, we will extend our demo example and discuss how to serialize and de-serialize list of Objects i.e.;

1. ArrayList of String Object
2. ArrayList of Custom Java object

Rule to serialize and de-serialize any object:

- Corresponsding class should implement *java.io.Serializable* interface
- For pre-defined in-built Java classes, it should be implementing *java.io.Serializable* interface

Exception: If we try to serialize any class that doesn't implement *java.io.Serializable* interface, then an exception will be thrown stating reason as "*NotSerializableException*"

So, for serializing ArrayList of String object –> both ArrayList and String should be serializable

- ArrayList is by default implements *java.io.Serializable* interface
- Also, String class implements *java.io.Serializable* interface

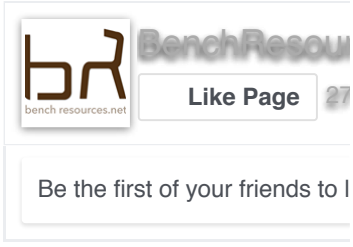
Let us focus on one simple Java program to serialize and de-serialize ArrayList of String objects

Serialization of ArrayList of String objects

SerializeArrayListOfStringObjects.java

```
1  package in.bench.resources.serialize.deserial
2
3  import java.io.FileNotFoundException;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.io.ObjectOutputStream;
7  import java.util.ArrayList;
8  import java.util.List;
9
10 public class SerializeArrayListOfStringObjects
11
12     public static void main(String[] args) {
13
```

Spring JDBC: An example on JdbcTemplate using Annotation
Java JDBC: An example to connect MS Access database
Oracle OSB 12c: Service Callout and Routing Table example
Oracle OSB 12c: Hello World service with both Business and Proxy Service



```
        // create ArrayList and inserts values
        List<String> leadersOfHistory = new Arr

        // add values to ArrayList
        leadersOfHistory.add("Joseph Stalin");
        leadersOfHistory.add("Adolf Hitler");
        leadersOfHistory.add("Benito Mussolini"
        leadersOfHistory.add("Napoléon Bonapart
        leadersOfHistory.add("Vladimir Putin");
        leadersOfHistory.add("Fidel Castro");
        leadersOfHistory.add("Robert Mugabe");


        // creating output stream variables
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;

        try {
            // for writing or saving binary dat
            fos = new FileOutputStream("SaveArr

            // converting java-object to binary
            oos = new ObjectOutputStream(fos);

            // writing or saving ArrayList valu
            oos.writeObject(leadersOfHistory);
            oos.flush();
            oos.close();
        }
        catch (FileNotFoundException fnfex) {
            fnfex.printStackTrace();
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }

        System.out.println("ArrayList object sa
    }
}
```

Output:

```
ArrayList object saved to SaveArrayList.ser fi
```

De-Serialization of ArrayList of String objects

Below program de-serializes “ArrayList of String objects”, which is serialized using above class

DeSerializeArrayListOfStringObjects.java

```
package in.bench.resources.serialize.deserial

import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;
import java.util.List;

public class DeSerializeArrayListOfStringObject

    public static void main(String[] args) {

        // creating input stream variables
        FileInputStream fis = null;
        ObjectInputStream ois = null;

        // creating List reference to hold AL v
        List<String> leadersOfHistory = null;

        try {
            // reading binary data
            fis = new FileInputStream("SaveArra

            // converting binary-data to java-o
            ois = new ObjectInputStream(fis);

            // reading object's value and casti
            leadersOfHistory = (ArrayList<Strin
        }
        catch (FileNotFoundException fnfex) {
            fnfex.printStackTrace();
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }
        catch (ClassNotFoundException ccex) {
            ccex.printStackTrace();
        }

        System.out.println("ArrayList object de

        // iterating & printing ArrayList value
        for(String leader : leadersOfHistory){
            System.out.println(leader);
        }
    }
}
```

Output:

```
ArrayList object de-serialized from SaveArrayL?;

Joseph Stalin
Adolf Hitler
Benito Mussolini
Napoléon Bonaparte
Vladimir Putin
Fidel Castro
Robert Mugabe
```

We will move on and see one more demo example on serializing and de-serializing ArrayList of custom Java object

What it means?

In above example, Serialization and De-serialization of ArrayList of String objects where both ArrayList and String classes are Serializable (by default)

But when we try to serialize and de-serialize ArrayList of custom Java object, then both ArrayList and custom Java object should be Serializable, otherwise "*NotSerializableException*" is thrown

By default, *ArrayList* implements *java.io.Serializable* interface. So, we need to implement *java.io.Serializable* interface for custom Java class

Let us move on and see demo example for serializing and de-serializing ArrayList of custom Java objects

Customer POJO with three member variables and their getter & setters

Customer.java

```
1  package in.bench.resources.serialize.deserial
2
3  import java.io.Serializable;
4
5  public class Customer implements Serializable {
6
7      // SerialVersionUID
8      private static final long serialVersionUID
9
10     // member variables
11     int customerId;
12     String customerName;
13     int customerAge;
14
15     // 3-arg parameterized constructor
16     public Customer(int customerId, String cust
17         this.customerId = customerId;
18         this.customerName = customerName;
19         this.customerAge = customerAge;
20     }
21
22     // overriding toString() method
23     @Override
24     public String toString() {
25         return "Customer [customerId=" + custom
26                 + " customerName=" + customerNa
27                 + " customerAge=" + customerAge
28                 + "]";
```

```
    }
}
```

SerializeArrayListOfCustomObjects.java

Below class serializes list of custom Java objects (i.e.; Customer class which is implementing *java.io.Serializable* interface)

```
package in.bench.resources.serialize.deserial
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

public class SerializeArrayListOfCustomObjects

    public static void main(String[] args) {

        // create List & ArrayList reference to
        List<Customer> listOfCustomers = new Ar

        // create customer objects using constr
        Customer napoleon = new Customer(1814,
        Customer mussolini = new Customer(1943,
        Customer hitler = new Customer(1945, "A
        Customer stalin = new Customer(1952, "J

        // add customer objects to ArrayList
        listOfCustomers.add(hitler);
        listOfCustomers.add(stalin);
        listOfCustomers.add(mussolini);
        listOfCustomers.add(napoleon);


        // creating output stream variables
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;

        try {
            // for writing or saving binary dat
            fos = new FileOutputStream("ArrayLi

            // converting java-object to binary
            oos = new ObjectOutputStream(fos);

            // writing or saving ArrayList valu
            oos.writeObject(listOfCustomers);
            oos.flush();
            oos.close();
        }
        catch (FileNotFoundException fnfex) {
            fnfex.printStackTrace();
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }
```

```
52
53          System.out.println("ArrayList of Custom
54                  + "ArrayListOfCustomer.ser file
55      }
56  }
```

Output:

```
1  ArrayList of Customer objects saved to ArrayLi
```

DeSerializeArrayListOfCustomObjects.java

This class de-serializes the list of custom Java objects (which is serialized from above class)

```
1   package in.bench.resources.serialize.deserial
2
3   import java.io.FileInputStream;
4   import java.io.FileNotFoundException;
5   import java.io.IOException;
6   import java.io.ObjectInputStream;
7   import java.util.ArrayList;
8   import java.util.List;
9
10  public class DeSerializeArrayListOfCustomObject
11
12      public static void main(String[] args) {
13
14          // creating input stream variables
15          FileInputStream fis = null;
16          ObjectInputStream ois = null;
17
18          // creating List reference to hold AL v
19          List<Customer> listOfCustomers = null;
20
21          try {
22              // reading binary data
23              fis = new FileInputStream("ArrayLis
24
25              // converting binary-data to java-o
26              ois = new ObjectInputStream(fis);
27
28              // reading object's value and casti
29              listOfCustomers = (ArrayList<Custom
30          }
31          catch (FileNotFoundException fnfex) {
32              fnfex.printStackTrace();
33          }
34          catch (IOException ioex) {
35              ioex.printStackTrace();
36          }
37          catch (ClassNotFoundException ccex) {
38              ccex.printStackTrace();
39          }
40
41          System.out.println("ArrayList object de
42                  + "ArrayListOfCustomer.ser file
43
```

```
44            // iterating & printing ArrayList value
45            for(Customer customer : listOfCustomers
46                System.out.println(customer);
47            }
48        }
49    }
```

Output:

```
1   ArrayList object de-serialized from ArrayListO
2
3   Customer [customerId=1945, customerName=Adolf Hi
4   Customer [customerId=1952, customerName=Joseph S
5   Customer [customerId=1943, customerName=Benito M
6   Customer [customerId=1814, customerName=Napoleon
```

Conclusion:

Thus, it is very easy to serialize and de-serialize any object in Java provided if it's corresponding class implements Serializable interface

References:

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html
https://docs.oracle.com/javase/7/docs/platform/serialization/spec/serial-arch.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html
https://docs.oracle.com/javase/tutorial/collections/intro/
https://docs.oracle.com/javase/tutorial/collections/interfaces/collection.html
https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html
https://docs.oracle.com/javase/tutorial/collections/interfaces/list.html
https://docs.oracle.com/javase/tutorial/collections/implementat

ions/list.html
https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html
https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Read Also:

- Java Serialization and De-Serialization Tutorial Index
- Serialization and De-Serialization in Java
- Serializable interface
- Transient keyword with Serialization in Java
- Transient keyword with static variable in Serialization
- Transient keyword with final variable in Serialization
- Serializing a variable with transient modifier or keyword
- Order of Serialization and De-Serialization
- Serialization with Aggregation
- Serialization with Inheritance
- Externalizable interface with example
- Serializable v/s Externalizable
- Importance of SerialVersionUID in Serialization
- Singleton Design pattern with Serialization
- How to stop Serialization in Java
- How to construct a singleton class in a multi-threaded environment in Java

Happy Coding !!
Happy Learning !!

« Serialization interview question and answer in Java

How to construct a singleton class in a multi-threaded environment in Java »

Related Posts:

1. **Serialization and De-Serialization in Java**
2. **Serializable interface**
3. **Serialization with Aggregation**
4. **Serialization with Inheritance**

ARRAYLIST DESERIALIZATION JAVA

LIST INTERFACE SERIALIZATION