CORE JAVA

" Google's Android Operating System is developed on Java platform. "

(http://www.addthis.com/bookmark.
us&s=linkedin&url=http%3A%2F%2Fw
deserialization.php&title=Serialization
/54c6bd7c7a894258/2&frommenu=1&

AdChoices ▶ Java Tutorials ▶ Java Test ▶ Java Example ▶ Core Java

(http://www.studytonight.com/)

ONLINE COURSES (http://studytonight.com/online-courses/)

LIBRARY (http://studytonight.com/library/)

STUDYROOM (http://studytonight.com/studyroom/)

FLASHCARDS NEW (http://studytonight.com/flashcards/)

Search

LogIn (http://www.studytonight.com/login)

SignUp (http://www.studytonight.com/register)

Suggest (http://www.studytonight.com/suggest)

Serialization and Deserialization in Java

Serialization is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or can be sent through streams. The reverse process of creating object from sequence of bytes is called **deserialization**.

A class must implement **Serializable** interface present in **java.io** package in order to serialize its object successfully. **Serializable** is a **marker interface** that adds serializable behaviour to the class implementing it.

Java provides **Serializable** API encapsulated under *java.io* package for serializing and deserializing objects which include,

- *java.io.serializable*
- *java.io.Externalizable*
- *ObjectInputStream*
- and *ObjectOutputStream* etc.

Marker interface

Marker Interface is a special interface in Java without any field and method. Marker interface is used to inform compiler that the class implementing it has some special behaviour or meaning. Some example of Marker interface are,

- java.io.Serializable
- java.lang.Cloneable
- java.rmi.Remote
- java.util.RandomAccess

All these interfaces does not have any method and field. They only add special behavior to the classes implementing them. However marker interfaces have been deprecated since Java 5, they were replaced by **Annotations**. Annotations are used in place of Marker Interface that play the exact same role as marker interfaces did before.

Signature of `writeObject()` and `readObject()`

writeObject() method of *ObjectOutputStream* class serializes an object and send it to the output stream.

```
public final void writeObject(object x) throws IOException
```

readObject() method of *ObjectInputStream* class references object out of stream and deserialize it.

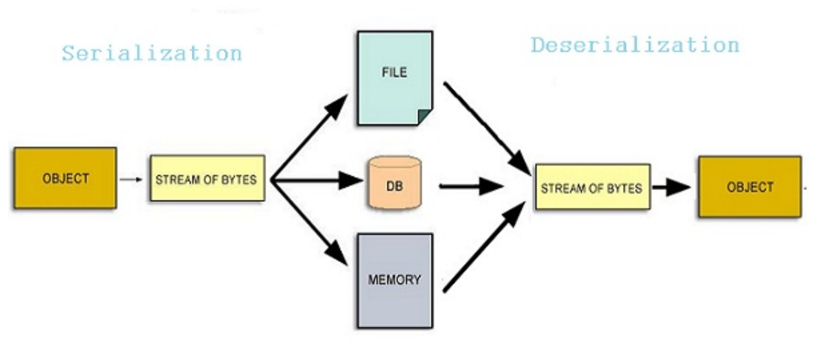
```
public final Object readObject() throws IOException,ClassNotFoundException
```

while serializing if you do not want any field to be part of object state then declare it either static or transient based on your need and it will not be included during java serialization process.

Test Yourself !

If you have studied all the lessons of JAVA, then evaluate yourself by taking these tests.

Topical Test (http://www.studytonight.com/java/tests/)



Serializing an Object

```
import java.io.*;
class studentinfo implements Serializable
{
 String name;
 int rid;
 static String contact;
 studentinfo(string n, int r, string c)
 {
  this.name = n;
  this.rid = r;
  this.contact = c;
 }
}

class Test
{
 public static void main(String[] args)
 {
 try
 {
  Studentinfo si = new studentinfo("Abhi", 104, "110044");
  FileOutputStream fos = new FileOutputStream("student.ser");
  Objectoutputstream oos = new ObjectOutputStream(fos);
  oos.writeObject(si);
  oos.close();
  fos.close();
  }
  catch (Exception e)
  { e. printStackTrace(); }
 }
}
```

Object of Studentinfo class is serialized using `writeObject()` method and written to **student.ser** file.

Deserialization of Object

```
import java.io * ;
class DeserializationTest
{
 public static void main(String[] args)
 {
  studentinfo si=null ;
  try
  {
   FileInputStream fis = new FileInputStream("student.ser");
   ObjectOutputStream ois = new ObjectOutputStream(fis);
   si = (studentinfo)ois.readObject();
  }
  catch (Exception e)
   { e.printStackTrace(); }
  System.out.println(si.name);
  System.out. println(si.rid);
  System.out.println(si.contact);
 }
}
```

```
Output :
Abhi
104
null
```

Contact field is null because,it was marked as static and as we have discussed earlier static fields does not get serialized.

NOTE : Static members are never serialized because they are connected to class not object of class.

transient Keyword

While serializing an object, if we don't want certain data member of the object to be serialized we can mention it transient. transient keyword will prevent that data member from being serialized.

```
class studentinfo implements Serializable
{
 String name;
 transient int rid;
 static String contact;
}
```

- Making a data member **transient** will prevent its serialization.
- In this example `rid` will not be serialized because it is **transient**, and `contact` will also remain unserialized because it is **static**.

← Prev (java-io-stream.php)   Next → (networking-in-java.php)