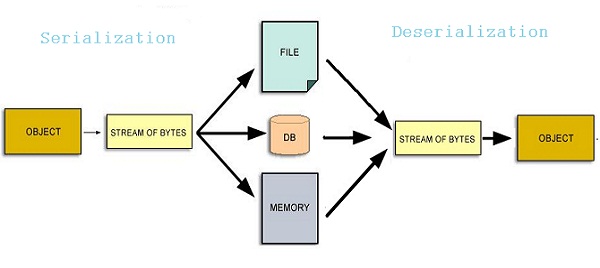
**Serialization and Deserialization in Java**

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

Serialization gives guarantee that when the object is retrieved back either from the file or network, the object maintains the same properties.



transient keyword in Java

**transient** is a variables modifier used in [serialization](http://quiz.geeksforgeeks.org/serialization-in-java/). At the time of serialization, if we don’t want to save value of a particular variable in a file, then we use **transient** keyword. When JVM comes across **transient**keyword, it ignores original value of the variable and save default value of that variable data type.

**transient** keyword plays an important role to meet security constraints. There are various real-life examples where we don’t want to save private data in file. Another use of **transient** keyword is not to serialize the variable whose value can be calculated/derived using other serialized objects or system such as age of a person, current date, etc.

To support serialization,**java.io** package comes with interface **Serializable**and two classes **ObjectOutputStream** and**ObjectInputStream**.

**1. class java.io.ObjectOutputStream**

Following is the class signature

**public class ObjectOutputStream extends OutputStream implements ObjectOutput, ObjectStreamConstants**

The methods of **ObjectOutputStream** are capable to save the state of primitive data types to an OutputStream. Only the serialized objects are allowed to write.

**Some write methods of ObjectOutputStream**

1. public void writeBoolean(boolean) throws java.io.IOException;
2. public void writeByte(int) throws java.io.IOException;
3. public void writeShort(int) throws java.io.IOException;
4. public void writeInt(int) throws java.io.IOException;
5. public void writeLong(long) throws java.io.IOException;
6. public void writeFloat(float) throws java.io.IOException;
7. public void writeDouble(double) throws java.io.IOException;
8. public void writeBytes(java.lang.String) throws java.io.IOException;
9. public void writeUTF(java.lang.String) throws java.io.IOException;
10. public final void writeObject(java.lang.Object) throws java.io.IOException;

**2. class java.io.ObjectInputStream**

Following is the class signature

**public class ObjectInputStream extends InputStream implements ObjectInput, ObjectStreamConstants**  
An **ObjectInputStream** is capable to **deserialize** primitive data and objects written earlier with**ObjectOutputStream**. Only serialized object (classes implementing java.io.Serializable or java.io.Externalizable) are allowed to read. **Reading the object and restoring the state is known as deserialization (reverse of Serialization)** and can done by the reading methods of ObjectInputStream.

An object written (to save the state) with **writeObject()** method of ObjectOutputStream can be read (otherway, restore the state) with **readObject()** method of ObjectInputStream.

Some reading methods of ObjectInputStream

1. public boolean readBoolean() throws java.io.IOException;
2. public byte readByte() throws java.io.IOException;
3. public char readChar() throws java.io.IOException;
4. public short readShort() throws java.io.IOException;
5. public int readInt() throws java.io.IOException;
6. public long readLong() throws java.io.IOException;
7. public float readFloat() throws java.io.IOException;
8. public double readDouble() throws java.io.IOException;
9. public java.lang.String readLine() throws java.io.IOException;
10. public java.lang.String readUTF() throws java.io.IOException;
11. public final java.lang.Object readObject() throws java.io.IOException, java.lang.ClassNotFoundException;

**3. interface java.io.Serializable (introduced with JDK 1.1)**

Following is the class signature

Compiled from “Serializable.java”  
**public interface java.io.Serializable {  
}**

Observe no methods in the interface.

The Serializable interface is very peculiar, in that, it does not contain any methods. The interface wihtout any methods is known as **Marker interface** or **Tag interface** or **Empty interface**

**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.

import java.io.\*;                          // for Serializable interface

public class Student implements Serializable

{

  int rollNumber;

  String name;

  double marks;

  transient String address;                // observe, transient keyword

                                           // constructor to assign properties

  public Student(int rollNumber, String name, double marks, String address)

  {

    this.rollNumber = rollNumber;

    this.name = name;

    this.marks = marks;

    this.address = address;

  }                                        // getter methods to retrieve properties

  public int getRollNumber( )

  {

    return rollNumber;

  }

  public String getName( )

  {

    return name;

  }

  public double getMarks( )

  {

    return marks;

  }

  public String getAddress( )

  {

    return address;

  }

}

import java.io.\*;

public class StudentWrite

{

  public static void main(String args[]) throws Exception

  {

    FileOutputStream fos = new FileOutputStream("abc.txt");

    ObjectOutputStream oos = new ObjectOutputStream(fos);

    Student std1 = new Student(35, "SNRao", 60.5, "Ameerpet, Hyderabad");

    Student std2 = new Student(45, "Jyostna", 70.5, "Paradise circle, Secunderabad");

    Student std3 = new Student(55, "Srinivas", 80.5, "Punjagutta, Hyderabad");

    oos.writeObject(std1);

    oos.writeObject(std2);

    oos.writeObject(std3);

                         // writing anonymous Student object

    oos.writeObject(new Student(65, "Jyothi", 90.5, "Kookatpally, Hyderabad"));

    fos.close();

    oos.close();

  }

}

===================================  
import java.io.\*;

public class StudentRead

{

  public static void main(String args[]) throws Exception

  {

    FileInputStream fis = new FileInputStream("abc.txt");

    ObjectInputStream ois = new ObjectInputStream(fis);

    while(true)

    {

      try

      {

        Student st = (Student) ois.readObject( );

        System.out.println(st.getRollNumber() + ", " + st.getName( ) + ", " + st.getMarks( ) + ", " + st.getAddress());

      }

      catch(EOFException e)

      {

        break;

      }

    }

    ois.close();

    fis.close();

  }

}