# Serialization and De-serialization in Java



**Serialization in Java** is a mechanism of *writing the state of an object into a byte-stream*.

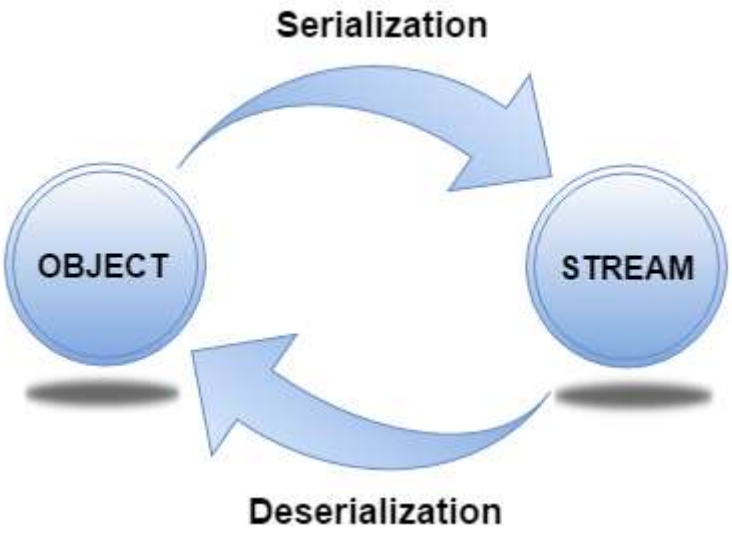
The reverse operation of serialization is called *deserialization* where byte-stream is converted into an object. The serialization and deserialization process is platform-independent, it means you can serialize an object in a platform and deserialize in different platform.

For serializing the object, we call the **writeObject()** method of *ObjectOutputStream*, and for deserialization we call the **readObject()** method of *ObjectInputStream* class.

We must implement the *Serializable* interface for serializing the object.

## Advantages of Java Serialization

It is mainly used to travel object's state on the network (which is known as marshaling).



# java.io.Serializable interface

Serializable is a marker interface (has no data member and method). It is used to "mark" Java classes so that the objects of these classes may get a certain capability.

It must be implemented by the class whose object you want to persist.

The String class and all the wrapper classes implement the *java.io.Serializable* interface by default.

Let's see the example given below:

```
1. import java.io.Serializable;
2. public class Student implements Serializable{
3.  int id;
4.  String name;
5.  public Student(int id, String name) {
6.   this.id = id;
7.   this.name = name;
8.  }
9. }
```

In the above example, Student class implements Serializable interface. Now its objects can be converted into stream.

# Example of Java Serialization

In this example, we are going to serialize the object of Student class. The writeObject() method of ObjectOutputStream class provides the functionality to serialize the object. We are saving the state of the object in the file named f.txt.

```
1. import java.io.*;
2. class Persist{
3.  public static void main(String args[]){
4.   try{
5.   //Creating the object
6.   Student s1 =new Student(211,"ravi");
7.   //Creating stream and writing the object
8.   FileOutputStream fout=new FileOutputStream("f.txt");
9.   ObjectOutputStream out=new ObjectOutputStream(fout);
10.  out.writeObject(s1);
```

```
11.  out.flush();
12.  //closing the stream
13.  out.close();
14.  System.out.println("success");
15.  }catch(Exception e){System.out.println(e);}
16. }
17.}
```

```
success
```

## Example of Java De-serialization

Deserialization is the process of reconstructing the object from the serialized state. It is the reverse operation of serialization. Let's see an example where we are reading the data from a deserialized object.

```
1.  import java.io.*;
2.  class Depersist{
3.   public static void main(String args[]){
4.    try{
5.    //Creating stream to read the object
6.    ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));

7.    Student s=(Student)in.readObject();
8.    //printing the data of the serialized object
9.    System.out.println(s.id+" "+s.name);
10.  //closing the stream
11.  in.close();
12.  }catch(Exception e){System.out.println(e);}
13. }
14.}
```

```
211 ravi
```