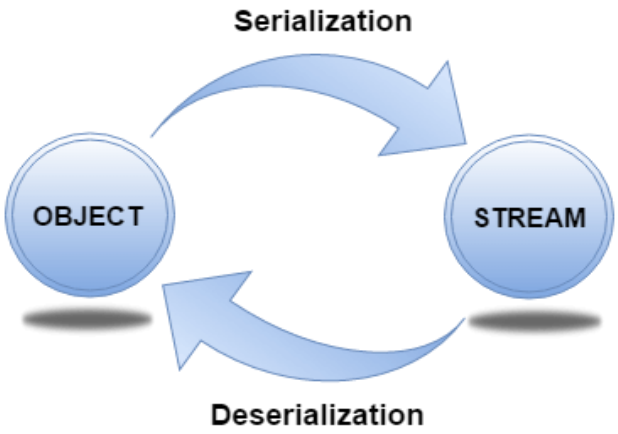
Serialization in Java is a mechanism of writing the state of an object into a byte stream.

It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called deserialization.

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 objects of these classes may get the certain capability. The Cloneable and Remote are also marker interfaces.

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:

```
import java.io.Serializable;

public class Student implements Serializable{

 int id;

 String name;

 public Student(int id, String name) {
```

```
  this.id = id;
  this.name = name;
 }
}
```

ObjectOutputStream class

The ObjectOutputStream class is used to write primitive data types, and Java objects to an OutputStream. Only objects that support the java.io.Serializable interface can be written to streams.

Constructor

1) public ObjectOutputStream(OutputStream out) throws IOException {}	creates an ObjectOutputStream that writes to the specified OutputStream.

Important Methods

Method	Description
1) public final void writeObject(Object obj) throws IOException {}	writes the specified object to the ObjectOutputStream.
2) public void flush() throws IOException {}	flushes the current output stream.
3) public void close() throws IOException {}	closes the current output stream.

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.

```
import java.io.*;
```

```
class Persist{

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

  Student s1 =new Student(211,"ravi");

  FileOutputStream fout=new FileOutputStream("f.txt");

  ObjectOutputStream out=new ObjectOutputStream(fout);

  out.writeObject(s1);

  out.flush();

  System.out.println("success");

 }

}
```

Deserialization in java

Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.

ObjectInputStream class

An ObjectInputStream deserializes objects and primitive data written using an ObjectOutputStream.

Constructor

1) public ObjectInputStream(InputStream in) throws IOException {}	creates an ObjectInputStream that reads from the specified InputStream.

Important Methods

Method	Description
1) public final Object readObject() throws	reads an object from the input stream.

IOException, ClassNotFoundException{}	
2) public void close() throws IOException {}	closes ObjectInputStream.

Example of Java Deserialization

```
import java.io.*;

class Depersist{

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

  ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));

  Student s=(Student)in.readObject();

  System.out.println(s.id+" "+s.name);

  in.close();

 }

}
```