ABOUT ARPIT MANDLIYA

I am java developer at Tata Consultancy Services Ltd. My current area of interest are J2EE,web development and java design patterns. I am technology enthusiast trying to explore new technologies. In spare time,I love blogging.

Serialization in java

Posted by: Arpit Mandliya in Core Java March 12th, 2013 18 Comments 1083 Views

Java provides mechanism called serialization to persists java objects in a form of ordered or sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. So if we have serialize any object then it can be read and deserialize it using object's type and other information so we can retrieve original object.

Classes ObjectInputStream and ObjectOutputStream are high-level streams that contain the methods for serializing and deserializing an object. ObjectOutputStream has many method for serializing object but commonly used method is:

```
1  private void writeObject(ObjectOutputStream os) throws IOException
2  {
3
4  }
```

Similarly ObjectInputStream has

```
1  private void readObject(ObjectInputStream is) throws IOException, ClassNotFoundException
2  {
3
4  }
```

Need of Serialization?

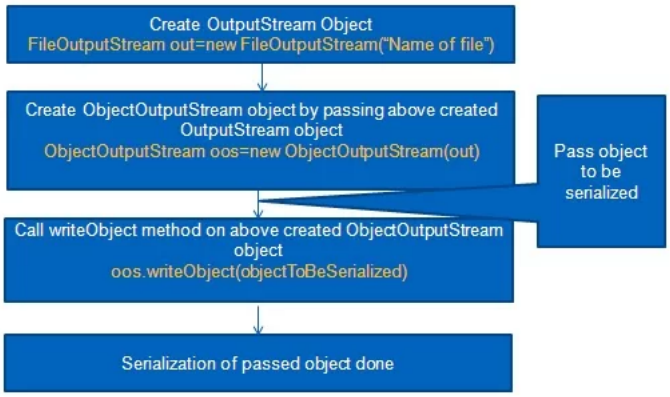
Serialization is usually used When the need arises to send your data over network or stored in files. By data I mean objects and not text. Now the problem is your Network infrastructure and your Hard disk are hardware components that understand bits and bytes but not Java objects. Serialization is the translation of your Java object's values/states to bytes to send it over network or save it.On other hand,Deserialization is conversion of byte code to corresponding java objects.

Concept of serialVersionUID :

SerialVersionUID is used to ensure that same object(That was used during Serialization) is loaded during Deserialization.serialVersionUID is used for version control of object.You can read more at serialVersionUID in java serialization

For Serialization:

Steps are :



Lets take an example: Create Employee.java in src->org.arpit.javapostsforlearning:

1.Employee.java

```
01  package org.arpit.javapostsforlearning;
02  import java.io.Serializable;
03  public class Employee implements Serializable{
04
05      int employeeId;
06      String employeeName;
07      String department;
08
09      public int getEmployeeId() {
10          return employeeId;
11      }
12      public void setEmployeeId(int employeeId) {
13          this.employeeId = employeeId;
14      }
15      public String getEmployeeName() {
16          return employeeName;
17      }
18      public void setEmployeeName(String employeeName) {
19          this.employeeName = employeeName;
20      }
21      public String getDepartment() {
22          return department;
23      }
24      public void setDepartment(String department) {
25          this.department = department;
26      }
27  }
```

As you can see above,if you want to serialize any class then **it must implement Serializable interface which is marker interface.**

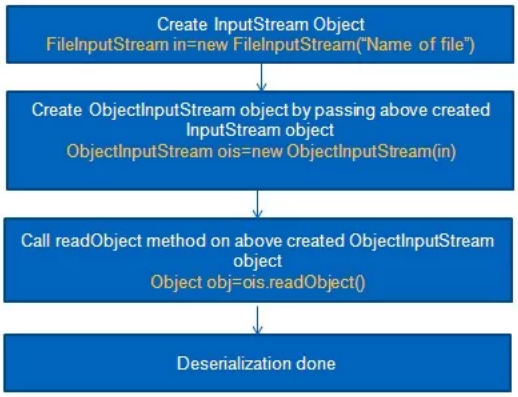
Marker interface in Java is interfaces with no field or methods or in simple word empty interface in java is called marker interface. Create SerializeMain.java in src->org.arpit.javapostsforlearning

2.SerializeMain.java

```
01  package org.arpit.javapostsforlearning;
02  import java.io.FileOutputStream;
03  import java.io.IOException;
04  import java.io.ObjectOutputStream;
05   public class SerializeMain {
06
07    /**
08     * @author Arpit Mandliya
09     */
10   public static void main(String[] args) {
11
12    Employee emp = new Employee();
13    emp.setEmployeeId(101);
14    emp.setEmployeeName('Arpit');
15    emp.setDepartment('CS');
16    try
17    {
18     FileOutputStream fileOut = new FileOutputStream('employee.ser');
19     ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
20     outStream.writeObject(emp);
21     outStream.close();
22     fileOut.close();
23    }catch(IOException i)
24    {
25     i.printStackTrace();
26    }
27   }
28  }
```

For Deserialization:

Steps are:



Create DeserializeMain.java in src->org.arpit.javapostsforlearning

3.DeserializeMain.java

```
01 package org.arpit.javapostsforlearning;
02 import java.io.IOException;
03 import java.io.ObjectInputStream;
04
05 public class DeserializeMain {
06  /**
07   * @author Arpit Mandliya
08   */
09  public static void main(String[] args) {
10
11   Employee emp = null;
12        try
13        {
14           FileInputStream fileIn =new FileInputStream('employee.ser');
15           ObjectInputStream in = new ObjectInputStream(fileIn);
16           emp = (Employee) in.readObject();
17           in.close();
18           fileIn.close();
19        }catch(IOException i)
20        {
21           i.printStackTrace();
22           return;
23        }catch(ClassNotFoundException c)
24        {
25           System.out.println('Employee class not found');
26           c.printStackTrace();
27           return;
28        }
29        System.out.println('Deserialized Employee...');
30        System.out.println('Emp id: ' + emp.getEmployeeId());
31        System.out.println('Name: ' + emp.getEmployeeName());
32        System.out.println('Department: ' + emp.getDepartment());
33  }
34 }
```

4.Run it:

First run SerializeMain.java then DeserializeMain.java and you will get following output:

```
1 Deserialized Employee...
2 Emp id: 101
3 Name: Arpit
4 Department: CS
```

So we have serialize an employee object and then deserialized it.It seems very simple but it can be very complex when reference object,inheritance come into the picture.So we will see different cases one by one and how we can apply serialization in different scenarios.

Case 1-What if an object has a reference to other objects

We have seen very simple case of serialization,now what if it also a reference to other objects.How will it serialized then? will reference object will also get serialized?.Yes,You don't have to explicitly serialize reference objects.When you serialize any object and if it contain any other object reference then Java serialization serialize that object's entire object graph.

For example:Lets say,Employee now has reference to address object and Address can have reference to some other object(e.g.Home) then when you serialize Employee object all other reference objects such as address and home will be automatically serialized. Lets create Address class and add object of Address as a reference to above employee class.

Employee.java:

```
01 package org.arpit.javapostsforlearning;
02 import java.io.Serializable;
03
04 public class Employee implements Serializable{
05
06  int employeeId;
07  String employeeName;
08  String department;
09  Address address;
10
11  public int getEmployeeId() {
12   return employeeId;
13  }
```

```
  public void setEmployeeId(int employeeId) {
   this.employeeId = employeeId;
  }
  public String getEmployeeName() {
   return employeeName;
  }
  public void setEmployeeName(String employeeName) {
   this.employeeName = employeeName;
  }
  public String getDepartment() {
   return department;
  }
  public void setDepartment(String department) {
   this.department = department;
  }
  public Address getAddress() {
   return address;
  }
  public void setAddress(Address address) {
   this.address = address;
  }
 }
```

Create Address.java in org.arpit.javapostsforlearning:

Address.java:

```
package org.arpit.javapostsforlearning;
public class Address {

 int homeNo;
 String street;
 String city;
 public Address(int homeNo, String street, String city) {
  super();
  this.homeNo = homeNo;
  this.street = street;
  this.city = city;
 }
 public int getHomeNo() {
  return homeNo;
 }
 public void setHomeNo(int homeNo) {
  this.homeNo = homeNo;
 }
 public String getStreet() {
  return street;
 }
 public void setStreet(String street) {
  this.street = street;
 }
 public String getCity() {
  return city;
 }
 public void setCity(String city) {
  this.city = city;
 }
}
```

Create SerializeDeserializeMain.java in org.arpit.javapostsforlearning:

SerializeDeserializeMain.java:

```
package org.arpit.javapostsforlearning;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializeDeserializeMain {
 /**
  * @author Arpit Mandliya
  */
 public static void main(String[] args) {

  Employee emp = new Employee();
  emp.setEmployeeId(101);
  emp.setEmployeeName('Arpit');
  emp.setDepartment('CS');
  Address address=new Address(88,'MG road','Pune');
  emp.setAddress(address);
  //Serialize
  try
  {
   FileOutputStream fileOut = new FileOutputStream('employee.ser');
   ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
   outStream.writeObject(emp);
   outStream.close();
   fileOut.close();
  }catch(IOException i)
  {
   i.printStackTrace();
  }

```

```
    //Deserialize
    emp = null;
    try
    {
     FileInputStream fileIn =new FileInputStream('employee.ser');
     ObjectInputStream in = new ObjectInputStream(fileIn);
     emp = (Employee) in.readObject();
     in.close();
     fileIn.close();
    }catch(IOException i)
    {
     i.printStackTrace();
     return;
    }catch(ClassNotFoundException c)
    {
     System.out.println('Employee class not found');
     c.printStackTrace();
     return;
    }
    System.out.println('Deserialized Employee...');
    System.out.println('Emp id: ' + emp.getEmployeeId());
    System.out.println('Name: ' + emp.getEmployeeName());
    System.out.println('Department: ' + emp.getDepartment());
    address=emp.getAddress();
    System.out.println('City :'+address.getCity());
   }
  }
```

Run it :

When you run SerializeDeserializeMain.java.You will get following output:

```
java.io.NotSerializableException: org.arpit.javapostsforlearning.Address
    at java.io.ObjectOutputStream.writeObject0(Unknown Source)
    at java.io.ObjectOutputStream.defaultWriteFields(Unknown Source)
    at java.io.ObjectOutputStream.writeSerialData(Unknown Source)
    at java.io.ObjectOutputStream.writeOrdinaryObject(Unknown Source)
    at java.io.ObjectOutputStream.writeObject0(Unknown Source)
    at java.io.ObjectOutputStream.writeObject(Unknown Source)
```

We got exception what went wrong.I forgot to mention,Address class must also be serializable.So you have to make Address serializable by implement serialzable interface.

Address.java:

```
import java.io.Serializable;

public class Address implements Serializable{

 int homeNo;
 String street;
 String city;
 public Address(int homeNo, String street, String city) {
  super();
  this.homeNo = homeNo;
  this.street = street;
  this.city = city;
 }
 public int getHomeNo() {
  return homeNo;
 }
 public void setHomeNo(int homeNo) {
  this.homeNo = homeNo;
 }
 public String getStreet() {
  return street;
 }
 public void setStreet(String street) {
  this.street = street;
 }
 public String getCity() {
  return city;
 }
 public void setCity(String city) {
  this.city = city;
 }
}
```

Run again:

When you run again SerializeDeserializeMain.java.You will get following output:

```
Deserialized Employee...
Emp id: 101
Name: Arpit
Department: CS
City: Pune
```

Case 2:What if you don't have access to reference object's source code(e.g you don't have access to above Address class)

If you don't have access to address class then how will you implement serializable interface in Address class.Is there any alternative to that? yes there is,You can create another class which extends address and make it serialzable but It can fails in many cases:

- What if class is declared as final
- What if class have reference to other non serializable object.

So then how will you serialize Employee object? so solution is you can make it transient.If you don't want to serialize any field then make it transient.

```
transient Address address
```

So after making address transient in Employee class when you run program.You will get nullPointerException because during deserialization address reference will be null

Case 3:What if you still want to save state of reference object(e.g above address object):

If you make address transient then during deserialization it will return null.But what if you still want to have same state as when you have serialized address object.Java serialization provides a mechnism such that if you have private methods with particular signature then they will get called during serialization and deserialization so we will override writeObject and readObject method of employee class and they will be called during serialization and deserialization of Employee object.

Employee.java:

```
package org.arpit.javapostsforlearning;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class Employee implements Serializable{

 int employeeId;
 String employeeName;
 String department;
 transient Address address;

 public int getEmployeeId() {
  return employeeId;
 }
 public void setEmployeeId(int employeeId) {
  this.employeeId = employeeId;
 }
 public String getEmployeeName() {
  return employeeName;
 }
 public void setEmployeeName(String employeeName) {
  this.employeeName = employeeName;
 }
 public String getDepartment() {
  return department;
 }
 public void setDepartment(String department) {
  this.department = department;
 }
 public Address getAddress() {
  return address;
 }
 public void setAddress(Address address) {
  this.address = address;
 }

 private void writeObject(ObjectOutputStream os) throws IOException, ClassNotFoundException
 {
  try {
   os.defaultWriteObject();
   os.writeInt(address.getHomeNo());
   os.writeObject(address.getStreet());
   os.writeObject(address.getCity());
  }
  catch (Exception e)
  { e.printStackTrace(); }
 }

 private void readObject(ObjectInputStream is) throws IOException, ClassNotFoundException
 {
  try {
   is.defaultReadObject();
   int homeNo=is.readInt();
   String street=(String) is.readObject();
   String city=(String) is.readObject();
   address=new Address(homeNo,street,city);

  } catch (Exception e) { e.printStackTrace(); }
 }
}
```

One thing should be kept in mind that ObjectInputStream should read data in same sequence in which we have written data to ObjectOutputStream. Create Address.java in org.arpit.javapostsforlearning:

Address.java:

```
package org.arpit.javapostsforlearning;
import java.io.Serializable;

public class Address {

```

```
  int homeNo;
  String street;
  String city;

  public Address(int homeNo, String street, String city) {
   super();
   this.homeNo = homeNo;
   this.street = street;
   this.city = city;
  }
  public int getHomeNo() {
   return homeNo;
  }
  public void setHomeNo(int homeNo) {
   this.homeNo = homeNo;
  }
  public String getStreet() {
   return street;
  }
  public void setStreet(String street) {
   this.street = street;
  }
  public String getCity() {
   return city;
  }
  public void setCity(String city) {
   this.city = city;
  }
 }
```

Create SerializeDeserializeMain.java in org.arpit.javapostsforlearning:

SerializeDeserializeMain.java:

```
package org.arpit.javapostsforlearning;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializeDeserializeMain {
 /**
  * @author Arpit Mandliya
  */
 public static void main(String[] args) {

  Employee emp = new Employee();
  emp.setEmployeeId(101);
  emp.setEmployeeName('Arpit');
  emp.setDepartment('CS');
  Address address=new Address(88,'MG road','Pune');
  emp.setAddress(address);
  //Serialize
  try
  {
   FileOutputStream fileOut = new FileOutputStream('employee.ser');
   ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
   outStream.writeObject(emp);
   outStream.close();
   fileOut.close();
  }catch(IOException i)
  {
   i.printStackTrace();
  }

  //Deserialize
  emp = null;
  try
  {
   FileInputStream fileIn =new FileInputStream('employee.ser');
   ObjectInputStream in = new ObjectInputStream(fileIn);
   emp = (Employee) in.readObject();
   in.close();
   fileIn.close();
  }catch(IOException i)
  {
   i.printStackTrace();
   return;
  }catch(ClassNotFoundException c)
  {
   System.out.println('Employee class not found');
   c.printStackTrace();
   return;
  }
  System.out.println('Deserialized Employee...');
  System.out.println('Emp id: ' + emp.getEmployeeId());
  System.out.println('Name: ' + emp.getEmployeeName());
  System.out.println('Department: ' + emp.getDepartment());
  address=emp.getAddress();
  System.out.println('City :'+address.getCity());
 }
}
```

Run it :

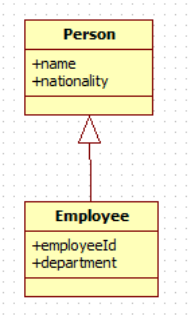
When you run SerializeDeserializeMain.java.You will get following output:

```
Deserialized Employee...
Emp id: 101
Name: Arpit
Department: CS
City :Pune
```

so now we got same state of address object as it was before serialization.

Inheritance in Serialization:

Now we will see how inheritance affects serialization.So there can be muliple cases whether super class is serializable or not.If not then how will you handle that and how it works.Lets see by example. We will create Person.java which will be superclass of Employee:



Case 4: What if superclass is Serializable?

If superclass is serialzable then all its subclasses are automatically serializable.

Case 5:What if superclass is not Serializable?

If super class is not serializable then we have to handle it quite differently.

- If superclass is not serializable then it must have no argument constructor.

Person.java

```
01  package org.arpit.javapostsforlearning;
02  public class Person {
03
04   String name='default';
05   String nationality;
06
07   public Person()
08   {
09    System.out.println('Person:Constructor');
10   }
11
12   public Person(String name, String nationality) {
13    super();
14    this.name = name;
15    this.nationality = nationality;
16   }
17
18   public String getName() {
19    return name;
20   }
21
22   public void setName(String name) {
23    this.name = name;
24   }
25
26   public String getNationality() {
27    return nationality;
28   }
29
30   public void setNationality(String nationality) {
31    this.nationality = nationality;
32   }
33
34  }
```

Create Employee.java in org.arpit.javapostsforlearning:

Employee.java:

```
01  package org.arpit.javapostsforlearning;
02  import java.io.Serializable;
03
04  public class Employee extends Person implements Serializable{
05
06   int employeeId;
07   String department;
08
09   public Employee(int employeeId,String name,String department,String nationality)
10   {
11    super(name,nationality);
12    this.employeeId=employeeId;
13    this.department=department;
14    System.out.println('Employee:Constructor');
15   }
16
17   public int getEmployeeId() {
18    return employeeId;
19   }
20   public void setEmployeeId(int employeeId) {
21    this.employeeId = employeeId;
22   }
23
24   public String getDepartment() {
25    return department;
26   }
```

```
 public void setDepartment(String department) {
  this.department = department;
 }
}
```

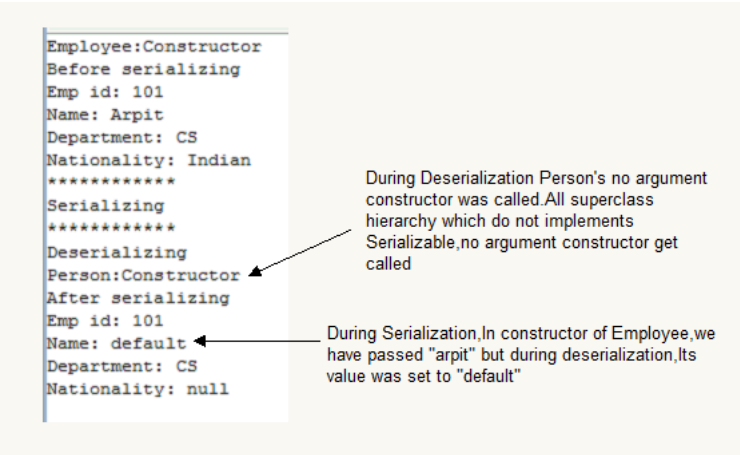
Create SerializeDeserializeMain.java in org.arpit.javapostsforlearning:

SerializeDeserializeMain.java:

```
package org.arpit.javapostsforlearning;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializeDeserializeMain {

 /**
  * @author Arpit Mandliya
  */
 public static void main(String[] args) {

  //Serialize
  Employee emp = new Employee(101,'Arpit','CS','Indian');
  System.out.println('Before serializing');
  System.out.println('Emp id: ' + emp.getEmployeeId());
  System.out.println('Name: ' + emp.getName());
  System.out.println('Department: ' + emp.getDepartment());
  System.out.println('Nationality: ' + emp.getNationality());
  System.out.println('************');
  System.out.println('Serializing');
  try
  {
   FileOutputStream fileOut = new FileOutputStream('employee.ser');
   ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
   outStream.writeObject(emp);
   outStream.close();
   fileOut.close();
  }catch(IOException i)
  {
   i.printStackTrace();
  }

  //Deserialize
  System.out.println('************');
  System.out.println('Deserializing');
  emp = null;
  try
  {
   FileInputStream fileIn =new FileInputStream('employee.ser');
   ObjectInputStream in = new ObjectInputStream(fileIn);
   emp = (Employee) in.readObject();
   in.close();
   fileIn.close();
  }catch(IOException i)
  {
   i.printStackTrace();
   return;
  }catch(ClassNotFoundException c)
  {
   System.out.println('Employee class not found');
   c.printStackTrace();
   return;
  }
  System.out.println('After serializing');
  System.out.println('Emp id: ' + emp.getEmployeeId());
  System.out.println('Name: ' + emp.getName());
  System.out.println('Department: ' + emp.getDepartment());
  System.out.println('Nationality: ' + emp.getNationality());
 }
}
```

Run it :

When you run SerializeDeserializeMain.java.You will get following output:



If superclass is not Serializable then all values of the instance variables inherited from super class will be initialized by calling constructor of Non-Serializable Super class during deserialization process. So here name is inherited from person so during deserialization,name is initialized to default.

Case 6-What if superclass is Serializable but you don’t want subclass to be Serializable

If you don’t want subclass to serializable then you need to implement writeObject() and readObject() method and need to throw NotSerializableException from this methods.

Case 7-Can you Serialize static variables?

No,you can’t.As you know static variable are at class level not at object level and you serialize a object so you can’t serialize static variables.

Summary:

- Serialization is the translation of your Java object’s values/states to bytes to send it over network or save it.On other hand,Deserialization is conversion of byte code to corresponding java objects.
- Good thing about Serialization is entire process is JVM independent, meaning an object can be serialized on one platform and deserialized on an entirely different platform.\

- If you want to serialize any class then it must implement Serializable interface which is marker interface.
- Marker interface in Java is interface with no field or methods or in simple word empty interface in java is called marker interface
- serialVersionUID is used to ensure that same object(That was used during Serialization) is loaded during Deserialization.serialVersionUID is used for version control of object.
- When you serialize any object and if it contain any other object reference then Java serialization serialize that object’s entire object graph.
- If you don’t want to serialize any field,then make it trasient.
- If superclass is Serializable then its subclasses are automatically Serializable.
- If superclass is not Serializable then all values of the instance variables inherited from super class will be initialized by calling constructor of Non-Serializable Super class during deserialization process.
- If you don’t want subclass to serializable then you need to implement writeObject() and readObject() method and need to throw NotSerializableException from this methods.
- You can’t serialize static variables.

Reference: Serialization in java from our JCG partner Arpit Mandliya at the Java frameworks and design patterns for beginners blog.

Tagged with: SERIALIZATION