



SERIALIZATION

CSI 211: OBJECT ORIENTED PROGRAMMING

Tanjina Helaly

SERIALIZATION

- **Serialization in java** is a mechanism of *writing the state of an object into a byte stream*.
- It is mainly used
 - to store/persist object's state
 - travel object's state on the network (known as marshaling).



DESERIALIZATION

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



STEPS FOR SERIALIZATION/DESERIALIZATION

- Mark the class Serializable
- Use ObjectOutputStream to serialize the object
- Use ObjectInputStream to deserialize the object



MARK THE CLASS SERIALIZABLE

- To make a object serializable, two conditions must be met:
 - The class must implement the `java.io.Serializable` interface.
 - Serializable is a marker interface (has no body). It is just used to "mark" java classes which support a certain capability.
 - All of the fields in the class must be serializable. If a field is not serializable, it must be marked **transient**.

- Example:

```
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;
    }
}
```



MARK THE CLASS SERIALIZABLE

- ▶ Once a class implements Serializable interface
 - all primitive attributes are marked as Serializable
 - To make the Reference type attributes serializable, those classes must be made serializable as well
 - **Note: All the objects within an object must be Serializable.**
 - Child class inherits the parent's property. So, all its sub classes will also be serializable.
 - If there is any static data member in a class, it will not be serialized because static is the part of class not object.
 - In case of array or collection, all the objects of array or collection must be serializable. If any object is not serializable, serialization will be failed.



CLASS USE FOR SERIALIZATION

- **ObjectOutputStream** - stream that contains the methods for serializing

Constructor

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

Important Methods

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



EXAMPLE CODE TO SERIALIZE

```
import java.io.*;

class Persist{

    public static void main(String args[]) {
        try{
            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");
        }
        Catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```



CLASS USE FOR DE-SERIALIZATION

- **ObjectInputStream** -stream that contains the methods for deserializing an object.

Constructor

1) <code>public ObjectInputStream(InputStream in) throws IOException {}</code>	creates an <code>ObjectInputStream</code> that reads from the specified <code>InputStream</code> .
--	--

Important Methods

Method	Description
1) <code>public final Object readObject() throws IOException, ClassNotFoundException {}</code>	reads an object from the input stream.
2) <code>public void close() throws IOException {}</code>	closes <code>ObjectInputStream</code> .



EXAMPLE CODE TO DESERIALIZE

```
import java.io.*;

class Depersist{

    public static void main(String args[]) {
        try{
            ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
            Student s=(Student)in.readObject();
            System.out.println(s.id+" "+s.name);
            in.close();
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```



JAVA SERIALIZATION WITH INHERITANCE

- Parent class properties are inherited to subclasses so if parent class is Serializable, subclass would also be.
- Now you can serialize the Student class object that extends the Person class which is Serializable.

Example

```
import java.io.Serializable;
class Person implements Serializable{
    int id;
    String name;
    Person(int id, String name) {
        this.id = id;
        this.name = name;
    }
}
```

```
class Student extends Person{
    String course;
    int fee;
    public Student(int id, String name, String course, int fee) {
        super(id,name);
        this.course=course;
        this.fee=fee;
    }
}
```

JAVA SERIALIZATION WITH AGGREGATION (HAS-A RELATIONSHIP)

- If a class has a reference of another class, all the references must be Serializable otherwise serialization process will not be performed. In such case, *NotSerializableException* is thrown at runtime.
- Since Address is not Serializable, you can not serialize the instance of Student class.

Example

```
class Address{  
    String addressLine,city,state;  
    public Address(String addressLine, String city, String state) {  
        this.addressLine=addressLine;  
        this.city=city;  
        this.state=state;  
    }  
}
```

```
import java.io.Serializable;  
public class Student implements Serializable{  
    int id;  
    String name;  
    Address address;//HAS-A  
    public Student(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
}
```

JAVA TRANSIENT KEYWORD

- ▶ If you **don't want to serialize** any data member of a class, you can mark it as transient.
- ▶ Let's say I have declared a class as Student, it has three data members id, name and age. If you serialize the object, all the values will be serialized but I don't want to serialize one value, e.g. age then we can declare the age data member as transient.
- ▶ If you deserialize the object, you will get the **default** value for transient variable.



EXAMPLE WITH TRANSIENT

```
import java.io.Serializable;
public class Student implements Serializable{
    int id;
    String name;
    transient int age;//Now it will not be serialized
    public Student(int id, String name,int age) {
        this.id = id;
        this.name = name;
        this.age=age;
    }
}
```



EXAMPLE WITH TRANSIENT

- ▶ Code to Serialize

```
import java.io.*;
class PersistExample{
    public static void main(String args[])throws Exception{
        Student s1=new Student(211,"ravi",22);//creating object
        //writing object into file
        FileOutputStream f=new FileOutputStream("f.txt");
        ObjectOutputStream out=new ObjectOutputStream(f);
        out.writeObject(s1);
        out.flush();

        out.close();
        f.close();
        System.out.println("success");
    }
}
```

- ▶ Output



EXAMPLE WITH TRANSIENT

▶ Code to De-Serialize

```
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+" "+s.age);
        in.close();
    }
}
```

▶ Output

- 211 ravi 0
- Notice age is set to 0 as it was marked transient.

