Q1: What is the purpose of StringTokenizer class ?

This class meant for tokenization purpose, present in java.util package. The default delimiter is space.

Ex:

StringTokenizer st=new StringTokenizer(“Ravi, Ramu, Rakesh”, ”,”);

While (st.hasMoreTokens())

{

System.out.println(st.nextToken());

}

Q2:What is Serialization?

The process of saving an object to file or the process of sending an object over network is called Serialization. But Strictly speaking, it is the process of converting an object from java supported form to file supported form or network supported form.

i.e Test.class------------->abc.ser

By using FileOutputStream & ObjectOutputStream classes, we can achieve serialization.

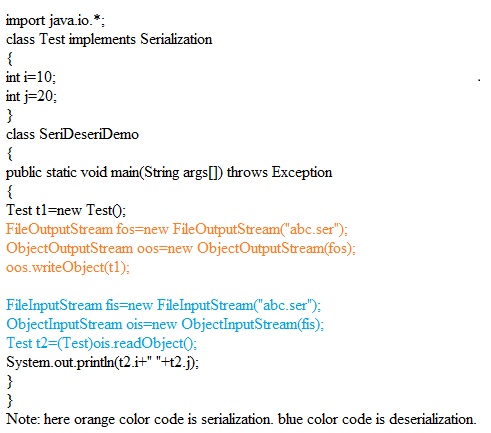
Q3:What is Deserialization?

The process of reading an object from a file or from the network is called Deserialization. But Strictly speaking, it is the process of converting an object from file supported form or network supported form to java supported form.

i.e abc.ser------------------> Test.class

By using FileInputStream & ObjectInputStream classes, we can achieve serialization.

See the following example for Serialization and Deserialization.



Q4:few important points about Serialization.

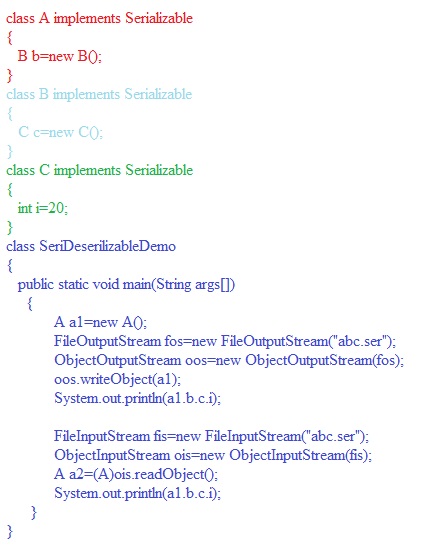
1. We can perform serialization for any Serializable object.
2. An object is said to be Serializable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package and doesn’t contain any method. It is a marker interface.
4. If we are trying to serialize any non serializable object, we will get NotSerializableException.
5. While performing serialization, if we don’t want to send the value of particular member variable in order to meet security constraint, such type of member variables we have to declare with transient keyword.
6. For the transient variable, jvm sends default value instead of original value. Transient means not to serializable.
7. static variables are not part of object state. They won’t participate in serialization process. Hence declaring static variable as transient doesn’t show any impact.
8. Final variable will be participate in the serialization directly by value. Hence declaring final variable as transient doesn’t show any impact.
9. see the following table carefully.

|  |  |
| --- | --- |
| Declaration | output |
| Int i=10; int j=20; | 10,20 |
| Int i=10; transient int j=20; | 10,0 |
| Int i=10; transient static int j=20; | 10,20 |
| Transient int i=10; Transient static int j=20; | 0,20 |
| Transient final int i=10; transient static int j=20; | 10,20 |

Q5:about object graphs in serialization?

1. Whenever we are saving an object to a file, the set of all its depended objects will be saved automatically. This set of objects is nothing but object graph.
2. In the object graph, every object should be serializable. Otherwise we will get NotSerializableException.

See the following example

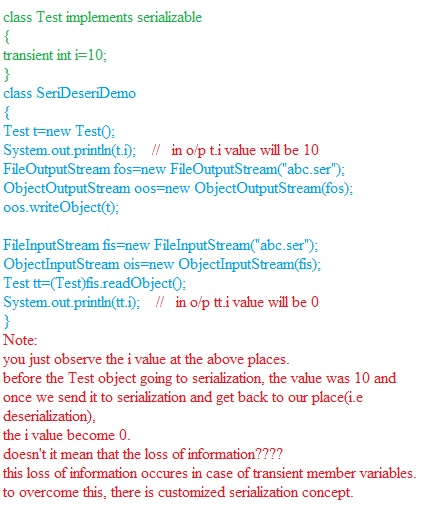


In the above program, Whenever we are trying to serialize A object, the B & C objects will be saved automatically. Because they are part of object graph of A object.

Among A,B,C classes, if any class is not serializable, we will get NotSerializableException. Hence every object in object graph should be serializable.

Q6: at what case we should go for customized serialization?

During serialization, there may be a chance of loss of information because of transient variable. For clarity see the following example.



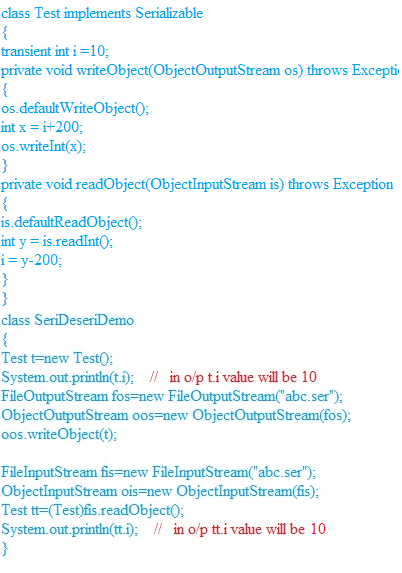
Q7: how to work with customized serialization?

To prevent the loss of information, we go for customized serialization.

We can achieve customized serialization by using the following two methods.

1. Private void writeObject(OutputStream os) this method will be executed automatically by the jvm, at the time of serialization.
2. Private void readObject(InputStream is) this method will be executed automatically by the jvm, at the time of deserialization.

See the following code to work with customized serialization



Q8:few points regarding serialization with respect to inheritance.

1. Serialiazation nature is inherited from parent to child i.e if the parent is serializable then automatically its child is also by default serializable.
2. Ex:

Class A implements Serializable

{

Int i=10;

}

Class B extends A

{

Int j=20;

}

Here, B can be serializable. Because B’s parent is already serializable.

Even though parent class is not serializable, we can serialize child class object individually.

T H E E N D