**Step 1 Choose Class that is to be serialized**

**First serialize class Student, which must implement the java.io.Serializable interface :**

|  |
| --- |
| **public** **class** Student **implements** java.io.Serializable  {  **public** String name;  **public** String address;  **public** **transient** **int** rollno;  **public** **int** roomNo;  } |

**Step 2 Instantiate a Student object and serialize it to a file.**

|  |
| --- |
| **import** java.io.\*;  **public** **class** SerializeExample {  **public** **static** **void** main(String[] args) {  Student e = **new** Student();  e.name = "Sammy Singh";  e.address = "3000 S. Federal";  e.rollno = 776778121;  e.roomNo = 118;  **try**  {  FileOutputStream fileOut = **new** FileOutputStream("student.ser");  ObjectOutputStream out = **new** ObjectOutputStream(fileOut);  out.writeObject(e);  out.close();  fileOut.close();  System.*out*.println("Object is serialized and stored in 'student.ser'");  }  **catch** (IOException ie) { ie.printStackTrace();  }  }  } |

**[ Output ]**

Object is serialized and stored in 'student.ser'

**Step 3 Deserialize the student object created in the above**

**(SerializeExample) code.**

|  |
| --- |
| **import** java.io.\*;  **public** **class** DeserializeExample {  **public** **static** **void** main(String[] args) {  Student e = **null**;  **try** {  FileInputStream fileIn = **new** FileInputStream("student.ser");  ObjectInputStream in = **new** ObjectInputStream(fileIn);  e = (Student) in.readObject();  in.close();  fileIn.close();  } **catch** (IOException i) {  i.printStackTrace();  **return**;  } **catch** (ClassNotFoundException c) {  System.*out*.println("Student class not found");  c.printStackTrace();  **return**;  }  System.*out*.println("Deserialized Student...");  System.*out*.println("Name: " + e.name);  System.*out*.println("Address: " + e.address);  System.*out*.println("Roll no: " + e.rollno);  System.*out*.println("Room No: " + e.roomNo);  }  } |

**[ Output ]**

Deserialized Student...

Name: Sammy Singh

Address: 3000 S. Federal

Roll no: 0

Room No: 118

**Serialization**



* **ObjectOutputStream & ObjectInputStream**
  + **Works like other input-output streams**
  + **They can write and read Objects.**
  + **ObjectOutputStream: Serializes Java Objects into a byte-encoded format, and writes them onto an OutputStream.**
  + **ObjectInputStream: Reads and reconstructs Java Objects from a byte-encoded format read from InputStream.**
* **Serialization can be used in.**
  + **Remote Method Invocation (RMI), communication between objects via sockets. (Marshaling and unmarshaling objects)**
  + **Archival of an object for use in a later invocation of the same program.**
* **Objects to be serialized** 
  + **Must implement *Serializable* interface**
  + **Non-persistent fields can be marked with *transient* keyword**
* **The following is written and read during serialization**
  + **Class of the object**
  + **Class signature**
  + **Values of all non-transient and non-static members**
* **To Write into an ObjectOutputStream**

**FileOutputStream out = new FileOutputStream(“afile”) ;  
 ObjectOutputStream oos = new ObjectOutputStream(out) ;  
 oos.writeObject(“Today”) ;  
 oos.writeObject(new Date()) ;  
 oos.flush() ;**

* **To Read from an ObjectInputStream**

**FileInputStream in = new FileInputStream(“afile”) ;  
 ObjectInputStream ois = new ObjectInputStream(in) ;  
 String today = (String) ois.readObject() ;  
 Date date = (Date) ois.readObject() ;**

* **ObjectOutputStream.writeObject(Object) traverses all the internal references of the object recursively and writes all of them.**
* **ObjectOutputStream implements DataOutput interface to write primitive data types.**

**writeInt(…), writeFloat(…), writeUTF(…), etc.**

* **ObjectInputStream implements DataInput interface ro read primitive data types.**

**readInt(), readFloat(), readUTF(), etc.**

* **writeObject(Object) throws NotSerializableException if Object does not implement Serializable interface**

**Externalizable interface**

* **To control the serialization process explicitly, Externalizable interface must be implemented.**
* **Externalizable interface**

**public interface Externalizable extends Serializable {**

**public void writeExternal(ObjectOutput out) throws IOException ;**

**public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException ;**

**}**

* **writeExternal and readExternal must save/load the state of the object. They must explicitly coordinate with its supertype to save its state.**

**java.io.Serializable interface:**

* **No methods**
* **Just a marker, indicating class is serializable**
* **Object / Class tree traversed to serialize**
* **java.io.NotSerializableException thrown, if unserializable member found**
* **Static and Transient members not serialized**

**import java.io.\*; Saver.java**

**import java.awt.\*;**

**public class Saver {**

**Saver() {**

**Button b = new Button("Bean Button");**

**b.setFont(new Font("System", Font.BOLD, 20));**

**try {**

**FileOutputStream f = new FileOutputStream ("Saver.ser");**

**ObjectOutputStream o = new ObjectOutputStream (f);**

**o.writeObject(b);**

**o.flush();**

**}catch (Exception e){ System.out.println(e); }**

**}**

**public static void main (String a[ ]) {**

**new Saver();**

**}**

**}**

**JTable Review**

**import** javax.swing.\*;

**import** javax.swing.table.\*;

**public** **class** InsertRows{

**public** **static** **void** main(String[] args) {

**new** InsertRows();

}

**public** InsertRows(){

JFrame frame = **new** JFrame("Inserting rows in the table!");

JPanel panel = **new** JPanel();

String data[][] = {{"Singh","100"},{"Peters","200"},{"Ronald","300"}};

String col[] = {"Name","code"};

DefaultTableModel model = **new** DefaultTableModel(data,col);

JTable table = **new** JTable(model);

//Insert first position

model.insertRow(0,**new** Object[]{"Dave","50"});

//Insert 4 position

model.insertRow(3,**new** Object[]{"Eddy","600"});

//Insert last position

model.insertRow(table.getRowCount(),**new** Object[]{"Frankie","600"});

panel.add(table);

frame.add(panel);

frame.setSize(300,300);

frame.setVisible(**true**);

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

}

}

