Slide 1

Centre International de Mathématiques Pures et Appliquées
International Center for Pure and Applied Mathematics

A CIMPA-UNSA-INRIA-UNESCO-VIETNAM SCHOOL ON

Objects and Network

Part : Persistence Courses

Emmanuel Chailloux
Emmanuel.Chailloux@lip6.fr
http://www-spi.lip6.fr/~emmanuel

Slide 2

SUMMARY

Persistence:

- Serialization in Java
- for threads
- for applets
- and using sockets

Slide 3

Persistence

To conserve an object outside of the current execution in a file or a stream.

Goal: to use the object in another program (or the same later)

Difficulties:

- to write and to read data structures
- including circular structures
- to typecheck during reading
- to save code

The jdk1.1 proposes a mechanism of **serialization**.

Slide 4

Serialization in Java

To serialize an object: just means to write the values of all its data fields into a stream of bytes.

- Stream classes
  - `ObjectOutputStream` : serialization stream
  - `ObjectInputStream` : deserialization stream
- interface
  - `Serializable` : must be implemented to be serializable

  It is an **empty** interface!!!

Slide 5

What is stored?

Storage contains :

- name and key (checksum) of the class
- all the values of all serializable data fields
- not the code, because the corresponding class must be existed

The key allows to check the version of the class.

The keyword `transient` marks data fields, which will not saved during serialization.

Another way: is to implement its own serialization mechanism. It's then mandatory to write a class **Externalizable** which implements the following interfaces : `readExternal` and `writeExternal`.

Slide 6

Example : Towns List

```
import java.io.*;

class ExTL implements Serializable {
  String nom;
  ExTL others;

  ExTL() {nom=null;others=null;}
  ExTL(String n, ExTL e) {
    nom=n;others=e;
  }

  boolean isEmpty() {return (nom == null);}
  void destroy() {nom=null;others=null;}

  public String toString() {
    if (isEmpty()) return "[]";
    else if (others.isEmpty()) return nom;
    else return nom+"::"+others.toString();
  }

  public void add (String unNom) {
    System.out.println("*");
    ExTL e = new ExTL(nom,others);
    others=e;
    nom=unNom;
  }

}
```

Slide 7

Example : Towns List (2)

```
import java.io.*;

class ExecuteTL {
  public static void main (String[] args) {
    ExTL e = new ExTL();
    ObjectOutputStream out;
    ObjectInputStream in;

   try {
    e.add("Hanoi");
    e.add("Paris");
    System.out.println("1 :  "+e);
    out = new ObjectOutputStream(new
FileOutputStream("ExTL.ser"));
    out.writeObject(e);
    out.flush(); out.close();
    e.destroy();
    System.out.println("2 :  "+e);
    in = new ObjectInputStream(new FileInputStream("ExTL.ser"));
    e.add("London");
    e.others = (ExTL)in.readObject();
    in.close();
    System.out.println("3 :  "+e);
    }
    catch (java.lang.ClassNotFoundException
exc){System.err.println(exc);}
    catch (StreamCorruptedException exc)
{System.err.println(exc);}
    catch (IOException exc) {System.err.println(exc);}
  }
}
```

Slide 8

Running TL

```
$ java ExecuteTL
*
*
1 : Paris::Hanoi
2 : []
*
3 : London::Paris::Hanoi
```

* : method `add(..)` is traced

Slide 9

Serialization and threads

A Thread is a class, so it can be serialized.

- can save a processus state
- and rerun after (in the same execution or not)
- needs to (re)initialize some value fields
  (particularlly to synchronize with others threads)
  for `transient` fields

Slide 10

To implement Serialize for a Consumer

```
class Consumer extends Thread implements Serializable {

  transient Shop aShop;
  String name;

  Consumer(Shop s, String sn){aShop=s; name = sn;}

  void changeShop(Shop s) {
    aShop = s;
  }
...
}
```

Slide 11

To save a Consumer

```
public class ProdConP {

  static void saveConsumer(Consumer c, String filename) {
    try {
      ObjectOutputStream out =
        new ObjectOutputStream(new FileOutputStream(filename));
      out.writeObject(c);
      out.close();
    }
    catch (Exception e){ System.out.println(e);}

  }
```

Slide 12

To load a Consumer

```
  static Consumer loadConsumer(String filename) {
      Consumer c = null;
      try {

      ObjectInputStream in =
        new ObjectInputStream(new FileInputStream(filename));
      c = (Consumer)in.readObject();
      in.close();
    }
    catch (Exception e) { System.out.println(e);}
    return c;
  }
...
}
```

Slide 13

To test the program

```
   Consumer c2 = new Consumer(myShop,"C12");
   c2.start();
   Consumer c3 = new Consumer(myShop,"C13");
   c3.start();

   c1.stop();
   char c;
   c=(char)System.in.read();
   switch (c) {
     case '1': { saveConsumer(c1,"C1.ser");break;}
     case '4': { Consumer cl = loadConsumer("C1.ser");
                 cl.changeShop(myShop);
                 cl.start();break;}
   }
```

Slide 14

Remarks

- Thread subclasses need to implement Serializable
- The `transient` field (as `Shop`) ares not serializable

After reading:

- It will need to initialize `transient` fields (`changeShop`)
- and to start the thread (`start`.

Slide 15

Serialization and applets

An Applet is a Thread, so it can be serialized.

- Component classes implement `Serializable`
  - including Panel class
  - and Event class
- It will need to include it inside the main applet :
  `add` and `setSize`
- and to initialize `transient` fields
- and finally to start it

Different behaviors between `appletviewer` and Java browsers.

Slide 16

Example : Game of Life

- Create a new Applet subclass which contains :
  - two TextField to give filename (load,save)
  - a GraphGOL field
- attach a method to each TextField in order to load and to save GraphGOL Applet
- after loading, start the new GraphGOL Applet

How can you create the first GraphGOL file?
⇒ by using the "save" menu in `appletviewer`

Slide 17

Execution : to save a GraphGOL

`save` object is a `TextField`
`gol` is an Applet.

```
String filename = (String)(save.getText()) +".ser";
System.out.println("Saving "+filename);
ObjectOutputStream out =
  new ObjectOutputStream (
    new FileOutputStream(filename));
out.writeObject(gol);
out.close();
```

Slide 18

Execution : to load an old GraphGOL

`load` is another `TextField`

```
String filename =  (String)(load.getText()) +".ser";
System.out.println("Loading "+filename);
ObjectInputStream in =
  new ObjectInputStream (
                  new FileInputStream(filename));
GraphGOL g = (GraphGOL)in.readObject();
in.close();
if (gol != null) {remove(gol);}
gol=g;
add("North",gol);
gol.initEvents();
gol.start();
```

Slide 19

Serialization and sockets

A Socket containts a Stream, so serializable Object can be sent to or received from a Socket!!!

- need to have same class in the server and in the client
- to allow to copy object between server and client :
  - to simplify communication protocol
  - to distribute objects (by copy)
- including threads and applets :
  - to allow to migrate threads
  - and to build applets server

Slide 20

Example 1 : GOL server

- A server keeps different serialized GOL object (in files).
- A client sends to the server a trivial request, which contains a specific filename.
- The server sends, by socket, the corresponding file (which contains a GOL object).

Slide 21

Example 2 : Consumer migration

- two shops servers run on different machines
- each server :
  - has a producer and some consumers
  - can accept connections to receive a consumer
  - can send a consumer to another server
- when a consumer migrates, it disappears from its server
- when a (new) consumer appears in a server, it must be synchronized to a local shop.

Slide 22

Remarks

no mobile code:

- classes must be known in the program
- there are no mobile code, only data fields are saved/loaded

for processus:

- threads and applets are objects and can implement `Serializable`, so they can be used this mechanism.

using sockets:

- socket streams can be used to send or to receive a `Serializable` object.
- protocols are simplified and will become the `Interface` object.
- exceptions can be send and throw inside the receiver.