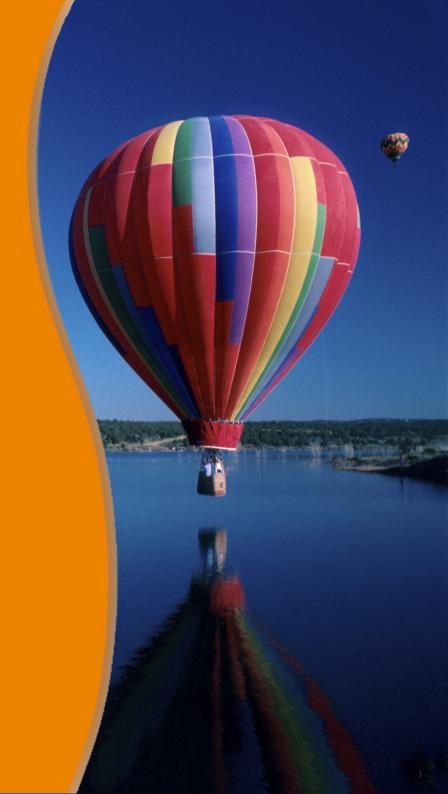
Serialization

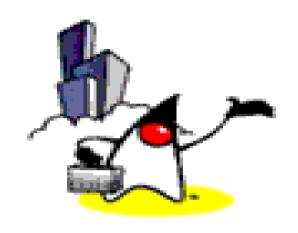
Sang Shin
Michèle Garoche
www.javapassion.com
"Learn with Passion!"



Topics

- What is Serialization?
- What is preserved when an object is serialized?
- Transient keyword
- Process of serialization
- Process of deserialization
- Version control
- Changing the default protocol
- Creating your own protocol via Externalizable





What is Serialization?

What is Serialization?

- Ability to read or write an object to a stream
 - Process of "saving" an object
- Used to save object to some permanent storage
 - Its state should be written in a serialized form to a file such that the object can be reconstructed at a later time from that file



Streams Used for Serialization

- ObjectOutputStream
 - For serializing (saving an object)
- ObjectInputStream
 - For deserializing (reconstructing an object)



Requirement for Serialization

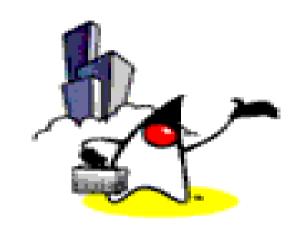
- To allow an object to be serializable:
 - Its class should implement the Serializable interface or inherit class that implements the Serializable interface
 - Serializable interface is marker interface (meaning it does not have any methods)
 - Its class should also provide a default constructor (a constructor with no arguments)



Non-Serializable Objects

- Most Java classes are serializable
- Objects of some system-level classes are not serializable
 - Because the data they represent constantly changes
 - Reconstructed object will contain different value anyway
 - For example, thread running in my JVM would be using my system's memory. Persisting it and trying to run it in your JVM would make no sense at all.
- A NotSerializableException is thrown if you try to serialize non-serializable objects



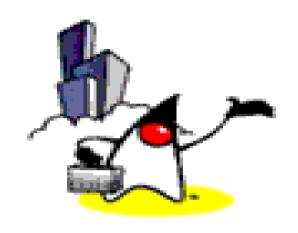


What is preserved when an Object is serialized?

What is preserved when an object is serialized?

- Enough information that is needed to reconstruct the object instance at a later time
 - Property values of the object are preserved
 - Methods and constructors are not part of the serialized stream
 - Class information is included, however





Process of Serialization

Serialization: Writing an Object Stream

 Use its writeObject method of the ObjectOutputStream class

 obj is the object to be written to the stream (meaning it is the object to be serialized)



Serialization: Writing an Object Stream

```
import java.io.*;
  public class SerializeBoolean {
     SerializeBoolean() {
3
        Boolean booleanData = new Boolean("true");
4
        try {
5
           FileOutputStream fos = new
6
                        FileOutputStream("boolean.ser");
7
           ObjectOutputStream oos = new
8
                        ObjectOutputStream(fos);
9
           oos.writeObject(booleanData);
10
           oos.close();
11
12 //continued...
```

Serialization: Writing an Object Stream





Process of Deserialization

 Use its readObject method of the ObjectInputStream class

```
public final Object readObject()
    throws IOException, ClassNotFoundException
```

 The Object type returned should be typecasted to the appropriate class before methods on that class can be executed



```
import java.io.*;
  public class UnserializeBoolean {
      UnserializeBoolean() {
3
        Boolean booleanData = null;
4
        try {
5
           FileInputStream fis = new
6
                         FileInputStream("boolean.ser");
7
           ObjectInputStream ois = new
8
                         ObjectInputStream(fis);
9
           booleanData = (Boolean) ois.readObject();
10
           ois.close();
11
12 //continued...
```

```
} catch (Exception e) {
13
           e.printStackTrace();
14
15
        System.out.println("Unserialized Boolean from "
16
                             + "boolean.ser");
17
        System.out.println("Boolean data: " +
18
                             booleanData);
19
        System.out.println("Compare data with true: " +
20
               booleanData.equals(new Boolean("true")));
21
22
23 //continued...
```



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

UnserializeBoolean usb =

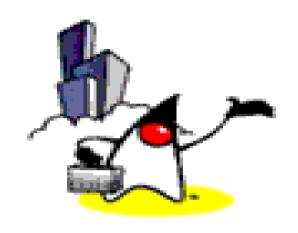
new UnserializeBoolean();
}
```



Demo:

SerializeAndDeserializeCurrentTime 1043_javase_serialization.zip





Transient keyword

When to use transient keyword?

- How do you serialize an object of a class that contains a non-serializable class as a field?
 - Like a Thread object
- What about a field that you don't want to to serialize?
 - Some fields that you want to recreate anyway
 - Performance reason
- Mark them with the transient keyword
 - The transient keyword prevents the field from being serialized
 - Serialization does not care about access modifiers such as *private*



Example: transient keyword

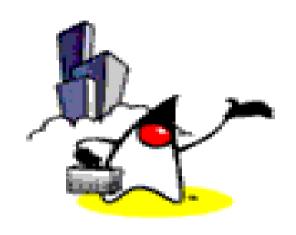
```
1 class MyClass implements Serializable {
2
     // Skip serialization of the transient field
     transient Thread thread:
4
     transient String fieldIdontwantSerialize;
5
6
     // Serialize the rest of the fields
7
     int data;
8
     String x;
9
10
     // More code
11
12 }
```



Demo:

SerializeAndDeserializeCurrentTimeTransient 1043 javase serialization.zip





Version Control

Version Control: Problem Scenario

- Imagine you create a class, instantiate it, and write it out to an object stream
- That saved object sits in the file system for some time
- Meanwhile, you update the class file, perhaps adding a new field
- What happens when you try to read in the flattened object?
 - An exception will be thrown -- specifically, the java.io.InvalidClassException
 - Why? (See next slide)



Unique Identifier

- Why exception is thrown?
 - Because all persistent-capable classes (*.class files) are automatically given a unique identifier
 - If the identifier of the class does not equal the identifier of the saved object, the exception will be thrown



Version Control: Problem Scenario Again

- However, if you really think about it, why should it be thrown just because I added a field? Couldn't the field just be set to its default value and then written out next time?
- Yes, but it takes a little code manipulation.
 The identifier that is part of all classes is maintained in a field called serialVersionUID.
- If you wish to control versioning, you simply have to provide the serialVersionUID field manually and ensure it is always the same, no matter what changes you make to the class file.

How Do I generate a Unique ID? Use serialver utility

- serialver utility is used to generate a unique ID
- Example

serialver MyClass

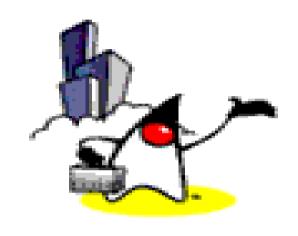
MyClass static final long serialVersionUID = 10275539472837495L;



Demo:

SerializeAndDeserializeCurrentTimeVersionControl 1043_javase_serialization.zip





Customizing the Default Protocol

Provide your own readObject() and writeObject() methods

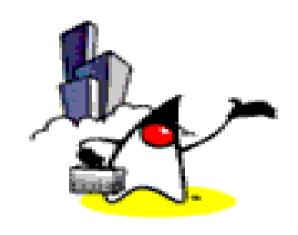
- Used when the default behavior of readObject() and writeObject() are not sufficient
- You provide your own readObject() and writeObject() in order to add custom behavior
- Example



Demo:

SerializeAnimationThreadNotStarted, SerializeAnimationThreadStarted 1043_javase_serialization.zip





Creating Your own Protocol via Externalizable interface

Externalizable Interface

- The writeExternal and readExternal methods of the Externalizable interface can be implemented by a class to give the class complete control over the format and contents of the stream for an object and its supertypes
- These methods must explicitly coordinate with the supertype to save its state
- These methods supersede customized implementations of writeObject and readObject methods



How does Object Serialization Scheme works with Externalizable

- Object Serialization uses the Serializable and Externalizable interfaces
- Each object to be stored is tested for the Externalizable interface
 - If the object supports Externalizable, the writeExternal method is called
 - If the object does not support Externalizable and does implement Serializable, the object is saved using ObjectOutputStream.





Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"

