Serialization

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 26, 2017

Overview I

Intro

- Provides the ability to save the state of an object beyond the life of the program and the virtual machine.
- Objects are flattened into byte code so they can be easily loaded later.
- Class to be serialized must implement the `Serializable` interface.

Default Serialization

- Can persist objects to a database using JDBC or across a network.
- **Cannot** persist static fields.
- `Object` doesn't implement `Serializable`.
- If an `Object` is `Serializable`, by default all fields are saved.
- When read in, the Constructor is not called.
- If class structure has changed, for example, fields or methods change, an `InvalidClassException` is thrown.
- Repeated writes do not overwrite previous writes, you must close and reopen to do so.
- By default if a class is `Iterable` it uses that to store the whole class.

Version Control

- All serialized classes contain a `serialVersionUID`.
- `serialVersionUID` is used by `readObject()` to check it's ok to load.
- If `serialVersionUID` is not provided, it is defaulted to a sum of hashCodes.
- If `serialVersionUID` is set, you can still load the .ser file even if fields have been added. If a conflict occurs and are incompatible, an exception is thrown.
- If `serialVersionUID` doesn't match that of the .ser file, then it won't be loaded.
- `transient` variables are not serialized with the object. It remains null until set in the loaded system.

Customised Serialization

- Easy to control how an `Object` is `serialized` by implementing `writeObject()` and `readObject()`.
- Still called as they are in the default manor, just the JVM checks if they are implemented or not and uses them if they are.
- They must be `private` so they cannot be overridden.

Example

```
public class HashMap<K,V> extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
{
transient Entry[] table; // Don't want to store the empty entries
private void writeObject(java.io.ObjectOutputStream s)
        throws IOException
    {
    // Write all the non-transient data using normal mehtod
       s.defaultWriteObject();
       s.writeInt(table.length); // Write number of buckets
       s.writeInt(size); // Write out number of mappings


       Iterator<Map.Entry<K,V>>
       // Returns set view of the map
       i = (size > 0) ? entrySet().iterator() : null;
       // Iterate over elements.
       if (i != null) {
              while (i.hasNext()) {
              Map.Entry<K,V> e = i.next();
              // Write objects
                     s.writeObject(e.getKey());
                     s.writeObject(e.getValue());
       }
```

Caching

- Cannot overwrite an object once saved without closing and opening again.
- Solution:
  1. Always close and repoen.
  2. Flush the cache by calling `out.reset()`.

The End