Serialisation

Josh Wilcox (jw14g24@soton.ac.uk)

2025-03-05

Contents

1 Object Serialisation

- Objects must be declared `serializable` to be serialised.
- This is done by implementing the interface `java.io.Serializable` in such class
  - `java.io.Serializable` is a 'tagging' **interface** because there are **no methods**
- Items that are declared serialisable can have their state recorded and output it on a datastream

1.1 Serialisation process

- Construct a `java.io.ObjectOutputStream`

```
ArrayList<Integer> myArray = new ArrayList<>();
myArray.add(4);
myArray.add(3);
myArray.add(2);
File f = new File("path/to/file");
OutputStream out = new FileOutputStream(f);
ObjectOutputStream oos = new ObjectOutputStream(out);

oos.writeObject(myArray);
oos.close();
```

2 Deserialisation

- Serialisation can be reversed to read objects from a data file
  - Must be declared serialisable again
- Similar process but we use a `ObjectInputStream` constructed with an `InputStream` object

3 Serialisation FAQ

- What has been preserved or saved?
  - **Only Properties** - Only member variables
  - All the objects share the same methods from the classes but may have different property values
  - So only properties actually need to be stored
- Will static properties be saved?
  - No, static properties are ignored - they will be sorted again as they are unique to each object
- Why cant we make all classes Serialisable?
  - Could cause problems ina new version of JDK in the future

4 Object Versioning

- Actual implementations of classes can vary over time
- Objects serialized a while ago may be saved from a different implementation of the class
- To solve this, a 64-bit secure hash of the full class description is written as part of its state - this is called its **version UID**

- During deserialisation, this UID is read and compared to existing classes.
  - If implementation has changed, the UIDs will not match and serialisation will fail

5 Serialisation of Reference Types

- When you call for serialisation of an object, JVM must serialise the whole object graph that is **heap reachable** from that object
- The graph structure should be preserved
- If an object has already been serialised and is referred somewhere else - it should not be serialised again

6 Externalisable Interface

- The serializable interface serialise all the **non-static** properties of the object
- Externalizable interface allows developers to **specify what to serialise**
  - Classes implementing will need a `public void WriteExternal(ObjectOutput out)` method
  - They will also need a `public void readExternal(ObjectInput in)` method
- Affords partial saving of objects

7 Transient Keyword

- During serialisation, if we **don't** want a particular attribute to be saved, then we can use the `transient` keyword

```
public class person implements Serializable{
    private transient String social_credit; // Social credit will never be
    ↪  saved
}
```

8 Default Serialisation

The Process:

- Write a hashcode of the current class object
- Serialise all fields of the object
  - Ignore statics
  - Ignore transients
  - Used fixed encoding for primitive types and String values
  - **Recursively** call `writeObject` on reference types - if they haven't been serialised already