Distributed Systems – Labs

File Handling and Object Serialisation in Java

Learning Outcomes:

1. Be able to use both byte oriented and character oriented streams in serialising data.
2. Be able to read and write to files, standard input and output and sockets.
3. Be able to use the various methods in the File class to examine and manipulate the file system.
4. Be able to write Java classes whose objects may be serialised.
5. Be able to serialise Java objects.
6. Be able to instantiate objects and access their state through their methods.

Tasks

Writing to Character Files

1. Extract `T1\FileTest1.java`. This program uses a character oriented stream to write simple lines of text to a file. Look up the PrintWriter class in the Java API docs to see what other methods are available. Modify your code to use some of these methods.

2. Extract T2\`FileTest2`.java. This program creates a character oriented reader from a byte oriented stream to read from standard input. It also creates a character oriented writer to write data to a file. Observe the use of various methods.

3. Extract `T3\FileTest3.java`. This program reads from the file created in step 2 above. Observe how string data is converted to integer data.

The `T3\Writer.java` prompts the user for input, and then writes the inputted data to a file.

The `T3\Reader.java` reads the contents of the file, and displays them.

Command Line Arguments

4. Extract `T4\Copy.java`. Note how this program allows a named file to be copied to another file using character oriented readers and writers.

5. Extract T5\`FileMethods.java`. Look up the File class in the Java API docs. Looking through the code, observe how the various methods can be used to traverse, analyse and modify the file system.

Object Serialisation

6. Extract T6\`Serialise.java`. Note how the class Personnel in this file implements the java.io.Serializable interface. Any objects that can be serialised by Java MUST implement this interface.

Note also how the use of `ObjectInputStream` and `ObjectOutputStream` can allow instances of the Personnel class to be serialised to a file.

7. Serialisable objects can be written to files as we have seen in step 6. However, they can also be written to any location for which we have an `OutputStream`, and read from anywhere we have an `InputStream`, such as our TCP sockets.

With this in mind, modify the `TCPEchoClient` and `TCPEchoServer` so that rather than writing text, they pass instances of Person objects, where Person objects contain name, age and address attributes. These objects should be created on the client side by the user entering required data, and should then be serialised and sent to the server which will print out the contents of the object.

ArrayLists

8. Extract T8\`ArrayListSerialise.java`

Note how this program creates three objects of class Personnel and then uses the add method of class ArrayList to place the objects into an ArrayList.

References

Most of today's lab came from Chapter 4 of Introduction to Network Programming in Java by Graba.

http://www.oracle.com/technetwork/articles/java/javaserial-1536170.html