File I/O: Serialization

20 Feb 2008
CMPT166
Dr. Sean Ho
Trinity Western University

(link: Sun Tech Guide on Serialization)

TRINITY WESTERN UNIVERSITY

java.io classes

- Byte-based streams:
  - FileInputStream, FileOutputStream
- Character-based streams:
  - FileReader, FileWriter
- Object-based streams:
  - ObjectInputStream, ObjectOutputStream
- Standard streams:
  - System.in, System.out, System.err
- Object holding pathname information:
  - File

Serializable objects

- Serialization is converting an object to a representation that can be written to a stream
- The Serializable interface is a tag:
  - Interface with no methods
  - Used to identify what objects are serializable
- Primitive types are serializable
- Arrays of serializable objects are serializable
- A class can be tagged as serializable if all its instance variables are serializable
    - Non-serializable vars can be declared transient (skipped during serialization)

Object-based I/O

- Use FileInputStream / FileOutputStream to open a file for binary I/O
  - fos = new FileOutputStream(“output.db”)
- Wrap the stream in an ObjectInputStream / ObjectOutputStream to use object serialization
  - oos = new ObjectOutputStream(fos);
- Use readObject/writeObject to do the I/O:
  - oos.writeObject(myobj);
  - readObject() returns a generic Object:
    - Cast it back to the original type
    - myobj = (MyObj) ios.readObject();

Customizing serialization

- Serializable objects: just tag as Serializable
  - all the work for how to read/write is done for you
- Methods writeObject() / readObject()
  - Specify exactly what format to use in writing out
  - Can call defaultWriteObject() to do the default functionality
  - Or use your own writeInt(), etc. to write out non-serializable fields

- See CustomDataExample.java

Random-access files

- Sequential files are hard to modify in-place
  - Must erase and rewrite entire file
- Random-access files:
    - file = new RandomAccessFile(“user.db”, “rw”);
- Can be used in place of FileInputStream / FileOutputStream, e.g., to do object-based I/O
- File position pointer:
    - file.seek(num_bytes);
  - Seek to position relative to start