

CSE 8B—Intro to CS: Java

Winter, 2006

February 7 (Day 9)
Binary Files

Instructor: Neil Rhodes

Binary files

The contents can't be opened in a text editor

Instead of string representations of numbers, etc, we have a duplicate of how it is represented in memory:

- Text file containing 3 shorts:
 - '1' '2' '3' '4' <newline> '5' '6' '7' '8' <newline> '5' '3' <newline>
- Binary file containing 3 shorts:
 - 48 57 22 46 0 53

2

Classes

DataInputStream (initialized with **InputStream**)

- Has methods to input different types:

- `int readInt()`
- `long readLong()`
- `short readShort()`
- `int readUnsignedByte();`
- `byte readByte();`
- `skipBytes(int n);`
- ...

DataOutputStream (initialized with **OutputStream**)

- Has methods to output different types

- `void writeInt(int i)`
- `void writeLong(long l)`
- `void writeShort(short s)`
- `void writeByte(byte b)`
- ...

3

Example: Reading tags from MP3 file

Format of tags in an MP3 file (simplified):

- ID3: the three characters 'I', 'D', '3'
- 2: version number. Others exist which mean different format of tag
- flags: 3 bytes
- Tag size: 3 bytes (each byte is 7 bits of the size)
- Array of frames
 - Frame type
 - 3 characters. For example, "TOA" = artist, "TAL" = album, "TT2" = title
 - frame size: 3 bytes
 - Encoding: 1 byte
 - Data
 - Usually a string. Any data after a null byte (0) should be ignored.

0000000	I	D	3	002	\0	\0	\0	\0	\b	w	T	T	2	\0	\0	@
0000020	\0	T	h	e		D	a	v	e		R	a	m	s	e	y
0000040	S	h	o	w		P	o	d	c	a	s	t	-			
0000060	F	r	i	,		0	2		D	e	c		2	0	0	5
0000100	2	3	:	5	9	:	0	0		-	0	5	0	0	\0	
0000120	T	A	L	\0	\0	036	\0	T	h	e		D	a	v	e	
0000140	R	a	m	s	e	y		S	h	o	w		P	o	d	c
0000160	a	s	t	\0	T	Y	E	\0	\0	006	\0	2	0	0	5	\0
0000200	T	D	R	\0	\0	026	\0	2	0	0	5	-	1	2	-	0

4

Part of Java program to dump MP3 Tags

```
import java.io.*;
class DumpMP3Tags {
    static DataInputStream f;

    public static void main(String[] args) {
        try {
            f = new DataInputStream(new FileInputStream(args[0]));
            if (f.readByte() == 'I' && f.readByte() == 'D' &&
                f.readByte() == '3' && f.readByte() == 2) {
                f.skipBytes(3);
                int size = readSize();
                int soFarInTag = 0;
                while (soFarInTag < size)
                    soFarInTag += readOneFrame();
            }
            f.close();
        } catch (IOException e) {
            System.err.println("exception: " + e.getMessage());
            System.exit(1);
        }
        ...
    }
}
```

5

Remainder of Java program to dump MP3 Tags

```
class DumpMP3Tags {
    .....
    public static int readSize() throws IOException{
        int b1 = f.readUnsignedByte();
        int b2 = f.readUnsignedByte();
        int b3 = f.readUnsignedByte();
        return b1 * 16384 + b2*128 + b3;
    }
    public static String readFixedLengthString(int length) throws IOException{
        String s = "";
        for (int i = 0; i < length; i++) {
            int b = f.readUnsignedByte();
            if (b != 0)
                s += (char) b;
        }
        return s;
    }
    public static int readOneFrame() throws IOException {
        String frameType = readFixedLengthString(3);
        f.skipBytes(1);
        int frameSize = f.readUnsignedShort();
        f.skipBytes(1); // skip encoding
        String frameContents = readFixedLengthString(frameSize-1);
        System.out.println(frameType + ": " + frameContents);
        return 6 + frameSize;
    }
}
```

6

Random-access files

Streams only provide one-way access

- No way to go back

Random-access means you can move around in the file

- Go to a certain location
- Where am I now?

RandomAccessFile

- RandomAccessFile(File f, String mode)
- RandomAccessFile(String f, String mode)
 - mode is "r" for read-only, "rw" for both read and write
- void writeInt(int i);
- int readInt();
- void seek(long location);
- long getFilePointer();
 - file pointer starts at beginning of file if file exists

7

Sometimes you want to write objects

ObjectInputStream

- Like DataInputStream but adds:
 - void writeObject(Object o)

ObjectOutputStream

- Like DataOutputStream but adds:
 - Object writeObject()

These can only read/write objects that are **serializable**

- class must implement serializable interface
 - class Foo implements serializable { ... }
- Primitive instance variables automatically read/written
- object instance variables also read/written using the same process
 - including arrays
- objects are written only once
 - If an ant and doodlebug both share a simulation, when re-read, they'll still share a simulation

8

Example reading/writing objects

Make Simulation serializable

- ```
public class Simulation implements Serializable {
```
- Make ants/doodlebugs/Rand/critters output
- ```
import java.io.*;
public class Rand implements Serializable {

import java.io.*;
public class Doodlebug extends Critter implements
Serializable {

import java.io.*;
public class Ant extends Critter implements Serializable

import java.io.*;
public abstract class Critter implements Serializable{
```
- Why not necessary to make Pair serializable?

9

Example reading/writing objects

When simulation is over, write it out, and read it back in:

```
s.writeSimulation("foo");
Simulation t = readSimulation("foo");
System.out.println(t.boardAsString());
}
public static Simulation readSimulation(String name) {
    Simulation t = null;
    try {
        ObjectInputStream f = new ObjectInputStream(new
FileInputStream(name));
        t = (Simulation) f.readObject();
        f.close();
    } catch (IOException e) {
        System.err.println("error: " + e.getMessage());
        System.exit(1);
    } catch (ClassNotFoundException e) {
        System.err.println("error: " + e.getMessage());
        System.exit(1);
    }
    return t;
}
```

10

Example reading/writing objects

When simulation is over, write it out, and read it back in:

```
s.writeSimulation("foo");
Simulation t = readSimulation("foo");
System.out.println(t.boardAsString());
}

public void writeSimulation(String name) {
    ObjectOutputStream f = null;
    try {
        f = new ObjectOutputStream(new FileOutputStream
(name));
        f.writeObject(this);
        f.close();
    } catch (IOException e) {
        System.err.println("error: " + e.getMessage());
        System.exit(1);
    }
}
```

11

How object streams handle duplicate objects

One object may refer to other objects

- Sometimes the same object more than once

Writing:

- First time it sees an object, it writes the object and a unique serial number
- Next time it sees that object, it writes just the serial number

Reading:

- When an object is read for the first time, it also reads the unique serial number
 - it saves away the object and serial number in an array
- When it reads just a serial number, it looks into the array for the corresponding already-existing object

12