ITEC 1630
Week 9: Files & Streams

Yves Lespérance

Readings: Horstmann Ch. 16

Using files

- We save data in files on disk or some other media so that we don't lose it even if the computer is shut off
- Files can store more data than may fit in working memory

Types of files

- Text files
- Binary files
  - Sequential access
  - Random access
  - Object streams

Text files

- Text files contain a sequence of characters.
- They are easy to understand for humans and can be read with a text editor
- The characters can only be read or written in sequence.

To read a text file

1. Open the file for reading by creating a FileReader: `FileReader r = new FileReader(`*`inputFileName`*`);` (may throw `FileNotFound` exception)
2. Create a Scanner for the reader: `Scanner s = new Scanner(r);`
3. Read and process the data using the scanner, e.g. `if(s.hasNextLine()){String l = s.nextLine();…}`
4. When finished, close the file: `r.close();`

To write a text file

1. Open the file for writing by creating a PrintWriter: `PrintWriter w = new PrintWriter(`*`outputFileName`*`);` if the file already exist, it will be overwritten
2. Write data to the file: `w.print(`*`value`*`);` or `w.println(`*`value`*`);`
3. When finished, close the file: `w.close();`

Can use JFileChooser dialog box to get the file name.

Can also use command line arguments.

Binary files

- Binary files contain a sequence of bytes in binary format; can represent any type of data.
- Usually a more compact representation than text.
- Can access data:
  - sequentially as a stream of bytes; low level
  - sequentially as an object stream; convenient
  - in arbitrary order as a `RandomAccessFile` of records

To read a binary file as a byte stream

1. Open the file for reading by creating a FileInputStream: `FileInputStream in = new FileInputStream(`*`inputFileName`*`);` (may throw `FileNotFound` exception)
2. Read and process the data:
```
while(!done)
{  int n = in.read(); // returns -1 when EOF
   if(n != -1){byte b = (byte) n;…}
   else {done = true;} }
```
3. When finished, close the file: `in.close();`

To write a binary file as a byte stream

1. Open the file for writing by creating an OutputStream: `OutputStream out = new OutputStream(outputFileName);` if the file already exist, it will be overwritten
2. Write data to the file: `out.write(byte);`
3. When finished, close the file: `w.close();`

To write a binary file as an object stream

1. Open the file for writing by creating an OutputStream and then an ObjectOutputStream: `ObjectOutputStream out = new ObjectOutputStream( new OutputStream(outputFileName));` if the file already exist, it will be overwritten
2. Write object(s) to the file: `out.writeObject(o);`
3. When finished, close the file: `w.close();`

Writing a binary file as an object stream

- Complete arrays or ArrayLists can be written as a single object
- Easiest to create an object that contains all your data and then write it to the object stream
- Objects written must implement Serializable interface (no methods required)
- If they contain non-serializable attributes, they are not automatically saved: declare these as `transcient` and define writeObject and readObject methods to handle them (see p. 599)

To read a binary file as an object stream

1. Open the file for reading by creating a FileInputStream and then an ObjectInputStream: `ObjectInputStream in = new ObjectInputStream( new FileInputStream(inputFileName));` (may throw `FileNotFound` exception)
2. Read object(s) from file, e.g. `BankAccount b = (BankAccount) in.readObject();`
3. When finished, close the file: `in.close();`

To read or write a binary file as a random access file

1. Decide on a record size and layout
2. Open the file for reading and writing by creating a RandomAccessFile: `RandomAccessFile f = new RandomAccessFile(fileName, “rw”);` for reading only use `“r”`
3. Move the file pointer to the right position: `f.seek(n * RECORD_SIZE);` or `f.seek(f.length());`
4. Write data to the file, e.g. `f.writeDouble(x)` or `f.writeInt(n)` or `f.writeChar(c);` or read data from the file, e.g. `Double d = f.readDouble()`
5. When finished, close the file: `f.close();`