Java

Text files

Lecture objectives

- To be able to use facilities of the File class to program the management of files.

- To be able to understand
  - Character streams
  - Reading from files –FileReader objects
  - Writing to files –FileWriter objects

I/O in Java

- I/O – Input/Output
  - Input to and output from programs
- Input can be from
  - Keyboards, mouse, scanners
  - Files
- Output cant be to
  - Monitors, printers
- Advantages of file I/O
  - Persistence
  - Pipping –output from one program can be input to another
  - Automation is simplified

I/O streams

- Stream
  - An object that either output data to its destination or obtains data from a destination
  - Plays the role of a buffer between source and destination
- Input stream
  - A stream that provides input for a program
  - System.in
- Output stream
  - A stream that accepts output from a program
  - System.out

Text files vs binary files

- Text files are readable by humans –and probably aliens...
  - Typically used for communications by humans
  - e.g. eclipse source files
- Binary files are more efficient though
  - Machines read& write binary files easily
  - Remember Java binary files are portable
    - Used on different machines
    - Reading binary files is done by programs
  - e.g. Executable files –.exe

Important concepts for I/O processing

- File paths
  - Locations on computer
- I/O buffering
  - Preventing I/O disk overhead
- Closing files
  - Conservation of resources
- Exception handling
  - A lot of things can go wrong

File paths

- Paths provide names for files and their locations on disk
  - Absolute paths
    - Full file paths
  - Relative paths
    - Relative to location of code/program
- Path names different for O/S
  - /home/phiri/projects/zcas/README.txt
  - C:\Users\My Documents\projects\zcas\README.txt
    - Escape special characters

I/O buffering

- Non-buffered
  - Bytes are read from/to disk instantly
    - Little delay between reads
    - A lot of disk overhead
- Buffered
  - Significant delay between reads
  - Significantly low disk overhead

Closing files

- An output file should be closed once writing process is done
- An input file should be close once reading process is done
- Use BufferedRead close() method
- When program terminates normally, any files opened are closed
- Why explicitly close files
  - In case of exceptions
  - In the event that a file previously opened for writing needs to be read

Exception handling

- I/O exceptions cannot be ignored
  - A number of things can go wrong
  - Non-existent file
  - Insufficient privileges to access file
  - Hardware errors
- IOException is a predefined class
  - Same drill
    - Wrap code in try block
    - Catch exceptions in catch block

Reading text files

- All code MUST be wrapped into a try block – remember unchecked and checked exceptions
- import java.io.* OR java.nio.*
- Create a file object
  - File f = new File(<location>);
- Create FileReader object using File object
  - FileReader fR = new FileReader(<file object>);
- Create BufferedReader object for buffering
  - BufferedReader bR = new BufferedReader(<file reader>);
- Read file

Reading text files

- Topic 10 Task 1

```
try {
 String line = "";
 File f = new File("/home/phiri/Sandbox/file1.txt");
 BufferedReader br = new BufferedReader(new   FileReader(f));
 while((line = br.readLine())!=null) {
            System.out.println(line);
      }
      br.close();
}
catch(IOException ioe) {
      ioe.printStackTrace();
}
```

Renaming text files

- Topic 10 Task 2

```
try {
        File f = new File("/home/phiri/Sandbox/201310_file1.txt");
        BufferedReader br = new  BufferedReader(new FileReader(f));
        if(f.renameTo(new File("/home/phiri/Sandbox/X201310_file1.txt"))) {
                System.out.println("File renamed!");
        }
        else {
                System.out.println("File NOT renamed!");
         }
}
catch(IOException e) {
        JOptionPane.showMessageDialog(null, "In go: " + e.toString());
}
```

Creating text files

- Topic 10 Task 3

```
try {
        int kIndex = 0;
        String wholeRecord = "";
        File f = new File("/home/phiri/Sandbox/201310_file2.txt");
        FileWriter fW = new FileWriter(f, false);
        BufferedWriter bW = new BufferedWriter(fW);
        PrintWriter pW = new PrintWriter(bW, true);
        Topic10Task3 t10T3 = new Topic10Task3(0, "", 0);
        wholeRecord = "XXXXX" + "\t" + "11111" + "\t" + "ZZZZZ";
        pW.println(wholeRecord);
        pW.close();
}
catch(Exception e) {
        JOptionPane.showMessageDialog(null, "In main: " + e.toString());
}
```

Renaming text files

- Topic 10 Task 2

```
try {
        File f = new File("/home/phiri/Sandbox/201310_file1.txt");
        BufferedReader br = new  BufferedReader(new FileReader(f));
        if(f.renameTo(new File("/home/phiri/Sandbox/X201310_file1.txt"))) {
                System.out.println("File renamed!");
        }
        else {
                System.out.println("File NOT renamed!");
         }
}
catch(IOException e) {
        JOptionPane.showMessageDialog(null, "In go: " + e.toString());
}
```

Directory listing

- Topic 10 Task 4a

```
try {
        File d = new File(".");
        if(d.isDirectory()) {
                File[] files = d.listFiles();
                for(File f: files) {
                        System.out.println(f.getName());
                }
        }
}
catch(Exception e) {
        JOptionPane.showMessageDialog(null, "In go: " + e.toString());
}
```

Deleting files

- Topic 10 Task 4b

```
try {
        File f = new File("/home/phiri/Sandbox/X201310_file1XXXXX.txt");
        if(f.delete()) {
                System.out.println("File deleted");
        }
        else {
                System.out.println("File Not deleted");
        }
}
catch(Exception e) {
        JOptionPane.showMessageDialog(null, "In go: " + e.toString());
}
```

Deleting files

- Topic 10 Task 5

```
try {
        String wholeRecord = "";
        File f = new File("fileInputx.txt");
        BufferedReader br = new BufferedReader(new FileReader(f));
        while((wholeRecord = br.readLine())!=null) {
                String[] data = wholeRecord.split("\t");
                int ref = Integer.parseInt(data[0]);
                String first = data[1];
                System.out.println(ref + " " + first);
        }
        br.close();
}
catch(Exception e) {
        JOptionPane.showMessageDialog(null, "In go: " + e.toString());
}
```

Lecture Outcomes

Today we have covered:

- Text files
  - Facilities for the File class
  - File management –renaming, creating& deleting files

- Questions?