**SOFT1002**

**Week 3: Files & Exception**

School of Information Technologies

---

**Today's Lecture**

- Reading and writing data to file
- Handling errors using Exception
- Converting string to number
- XML
- Read Big Java: Ch.15.1-2, 14

---

### Announcement

- **Task A Stage 1: Individual Certification**
  - Due next week (Week 4) in the lab, **however**
  - If your lab is on Monday,
    - Demo your codes according to the test plan on Monday
    - Submit your codes (including vulnerability) and test plan to your tutors during the tutorial session
  - A test which only shows menu interface is not acceptable. You need to write 2 classes that have inheritance and/or file features.

---

### Issues with Task A Stage 1

- Remember to read the marking criteria for Task A Stage 1.
- No interactive input during demo; test cases should be encoded in the Tester class.
- Each student should have a test plan for the classes they developed.

---

### Test Plan

<table>
<thead>
<tr>
<th>Objective</th>
<th>Input</th>
<th>Expected Output</th>
<th>Actual Output</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the program can handle an empty file.</td>
<td>An empty file.</td>
<td>Message &quot;empty input file&quot; displayed on screen.</td>
<td>Message &quot;empty input file&quot; displayed on screen.</td>
<td>Done (14.1.3.2)</td>
</tr>
<tr>
<td>Test the program is able to write data to file.</td>
<td>&quot;John&quot; and &quot;SOFT1002&quot; written as two separate lines to the file.</td>
<td>&quot;John&quot; and &quot;SOFT1002&quot; in the file.</td>
<td>No data found in the file.</td>
<td>Correction required. Forgot to push the content in buffer to file.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

### File I/O in Java: Everything is done by Streams

```
FileReader attached to file, keyboard, network
```

```
variable
```

- buffer
- character
- object
BufferedReader

- From file
  - br = new BufferedReader(new FileReader(fileName));
- From keyboard
  - br = new BufferedReader(new InputStreamReader(System.in));
- From network
  - br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

Serialization - What is it?

- Rather than saving the text representation of an object, what if you can read and write objects whole?
  - This can be called serialization in Java
- Using serialization is fairly trivial
  - Objects which you wish to serialize must be instantiated from classes which implement the Serializable interface
  - To read objects, use ObjectInputStream
  - To write objects, use ObjectOutputStream
  - If you do not want a particular instance variable to be serialized, use the keyword transient

Exception

- Exceptions are an error handling mechanism for run time error
- Java forces you to handle some errors by using Exception, e.g. manipulating file objects
- If there is no value you can sensibly return to indicate an error has occurred, throw an exception
- You can create your own Exception classes

Exception – Java Syntax

```java
try {
    // throw region
} catch (exception declaration 1) {
    // catch region
}
catch (exception declaration 2) {
    // catch region
}
finally {
    // finally block is optional
}
```

Why would I throw an exception?

- You throw an exception, rather than catching it, when:
  - It does not make sense for you to handle the error yourself (by catching the exception)
  - The same type of error happens in several different locations in a program and you want to localize the handling of the error
- Note: Eventually, someone must catch the exception, otherwise, it will come up as a run time error

Markup Languages

- A markup language
  - Uses tag to give additional information to the user’s contents. Just like we use a double-quote to start and end someone’s speech.
  - Tags are in enclosed by angular brackets
    - Example: <familyName>Smith</familyName>
  - SGML (Structured document Markup Language) has been used by the publishing industry for a long time
- HTML (Hyper Text Markup Language):
  - Derived from SGML
    - An SGML application is a language specified with an SGML grammar.
  - Widespread in WWW
  - The tags are for content description and formatting purposes
    - Content Description: <header>, <title>
    - Formatting: <br>, <i>
**XML: eXtensible Markup Language**

- A language for defining markup languages
- Motivation: To separate content/data from presentation. XML documents store content while stylesheets store presentation.
- Other advantage: The document is in text-format, hence, you can read the content without the need of any proprietary software.
- You can even define grammar for your data content. Two kinds of XML grammars are:
  - DTD – document type definition, the original
  - XML Schema, the latest

**XML: a sample**

An XML document is a text file:

```xml
<xml version="1.0" standalone="yes" encoding="UTF-8"/>
<sampleDoc>
  <title>The meaning of it all</title>
  <body>From the first human civilisations ...
</sampleDoc>
```

1. It contains a nested tree of elements.
2. Each element has start and end tags e.g. `<title>` and `</title>`.
3. Elements can contain text between the start and end tags, which we call its content e.g. "The...all" is the content of the title element.
4. Elements can nest e.g. the sampleDoc element contains title and body elements.
5. An XML document that follows these rules is called well formed.
6. Valid: It is not only well-formed, but it also agrees with the XML grammars.

**Well-formed XML**

Make sure you include start and end tag each time you write your contents to file in your TaskA and check this constraint when you read the data from file.

**Correct**

```xml
<dog>
  <name>Lucky</name>
  <age>3</age>
</dog>
```

**Incorrect**

```xml
<turtle>
  <name>Snow</name>
  <age>3</ages>
</turtle>
```

**An Example**

Connected to XML, it looks like:

```xml
<database>
  <customer>
    <id>0001</id>
    <name>Kingston</name>
    <address>school of IT</address>
    <address>The University of Sydney</address>
  </customer>
</database>
```

**What have we done?**

- Reading and writing data to file
- Handling errors using Exception, `cf. throw`, `throws`, `catch`, `finally`
- Converting string to number
  - `cf Integer.parseInt(number)` and `Double.parseDouble(number)`
- Prepare for next week:
  - SOFT1002 Resource Book: Ch.16 of the excerpts from the Kingston’s book.

**Resources for XML**

- Websites:
  - www.w3.org/XML/
  - www.xml.org
  - www.oasis-open.org
  - xml.coverpages.org/xml/
  - http://www.ucc.ie:8080/cocon/XMLFAQ

- Tutorial