Australian National University

make

Eric McCreath

Overview

In this lecture we will:

- introduce the idea of automatic build systems
- overview the **make** command and how it works
- learn about the basics of the format of a **Makefile**
- look at some Makefiles for Java projects, and
- examine some more advanced options

Build Management Tools

As programs grow larger, compliing them often becomes more complicated. As a programmer, attempt to avoid:

- typing the commands each time (which is time consuming and you may miss compiling changed source),
- have a script that compiles everything (for big programs this can make compiling a very slow process).

If libraries or compiler options are used when compiling code, it is good practice to record these for future reference.

For small, single file programs a simple way of documenting how to compile the code is to include the compile command as a comment in the source. However, for larger codes a build management tool must be used.

Build Management Tools

The *make* command is old, simple and widely used. However, other popular tools include:

- **ant** - used widely for Java. Uses XML files for describing requirements
- **eclipse** - has inbuilt tools for compiling code
- **maven** - is targeted at Java and uses XML although it has more default behavior based on available files
- **configure** - used for generating Makefiles for different systems
- **gradle** - build and automation tool.
- **jenkins** - a continuous integration tool run on a server. Usually configured to do nightly compiles/builds and perform various tests.

make

The *make* command, when executed, looks for the *Makefile* (or *makefile*) in the current directory.

This file describes:

- the commands used to compile the code
- what source files need compiling
- the dependencies between the source files.

Australian National University

make

- *make* uses the dependency information in the *Makefile* along with the modification times on those files to determine which commands to execute.

- The disadvantage of this approach is that a small change in one file can trigger an avalanche of dependencies which may require the entire code base to be re-compiled.

- The *make* command is targeted at compiling code. However, it can be used in other situations when commands need to be executed based on modification of files. For e.g. the latex document system, or static html generated from scripts

Makefile

The *Makefile* is just a plain text file that is formatted in a very particular way.

- it is made up of a sequence of *rules*
- each rule has a *target* (or targets)
- rules have a list of dependencies (which are either files or other targets)
- rules are followed by a sequence of commands that would normally, based on the dependencies, generate (or achieve) the target
- anything on a line after a # is just a comment.

Makefile

```
# rule 1
target1 : dep1 dep2 dep3
	command1
	command2
# rule 2
target2 : dep4 dep5
	command3
```

- IMPORTANT - **the Makefile format requires a *tab* before the command**
- Also notice the ":" after the targets.

Simple Example

- If the text below was saved in a Makefile within a directory containing "Hello.java", the *make* command could be used to build, run and clean up the code.

```
Hello.class : Hello.java
    javac Hello.java

run : Hello.class
    java Hello

clean :
    rm Hello.class
```

- If *make* is run with no parameters then the target of the first rule is the target that will be executed

- *make* can also be executed with a target as a parameter. For e.g. in this case *make run* or *make clean*

Macros

Often some text within a Makefile is repeated. One might want to have a *variable* which is modified at one point in the Makefile and re-used in multiple places within the file. *Macros* can be used for this purpose. To define a macro:

```
MACRONAME = macrostring
```

It is used as $(MACRONAME). An example Makefile is:

```
OPTIONS = -cp bin:. -g
Hello.class : Hello.java
    javac $(OPTIONS) Hello.java
```

Macro instantiation is lazy. They are only expanded when used. There are a number of useful predefined macros including:

```
$? - the dependencies of the rule being executed
$< - the first dependency
$@ - the target being executed
```

Australian National University

Patterns

Often the same set of commands need to be executed on the same type of files.

Rather than specify the actual filenames, a *pattern* representing the filenames can be specified.

The rule below would compile C code (files with suffixes .c) into their respective object binaries (files with suffixes .o)

```
%.o : %.c
    gcc -c $< -o $@
```

Exam Questions - Examples

- What are some reasons for using *build management tools* ?
- Explain how the *make* command determines which commands to execute.
- Suppose you had a project that consisted of the following Java source files: MainGUI.java, WorldState.java, and OtherInfo.java. With the MainGUI.java containing the main method, construct a simple Makefile that would enable you to compile (using just "make") and run (using just "make run") this project.

Australian National University

References

Wikipedia has a few useful pages on this topic:

```
http://en.wikipedia.org/wiki/Make_%28software%29#Modern_versions
http://en.wikipedia.org/wiki/Build_automation
```

The GNU make manual:

```
http://www.gnu.org/software/make/manual/make.html
```

Or just *man* make:

```
% man make
```

Australian National University

Persistent Data Bespoke and Serializable

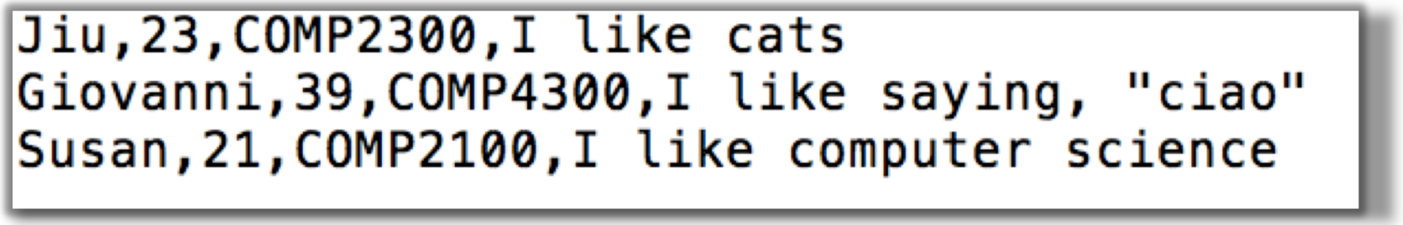
Michael Curtotti
Adapted from lectures by
Eric McCreath

Overview

- Briefly discuss bespoke and serialization as approaches to data persistence
- Demonstrate the approaches by looking at a couple of simple programs

Bespoke - Example

- We are going to use a simple comma delimited text file to store Student data, Each student has a name, age, course and preference.
- We will create Student objects using a Student class and when loading the data into the program store each student in a Hashmap – we will save to a simple text file



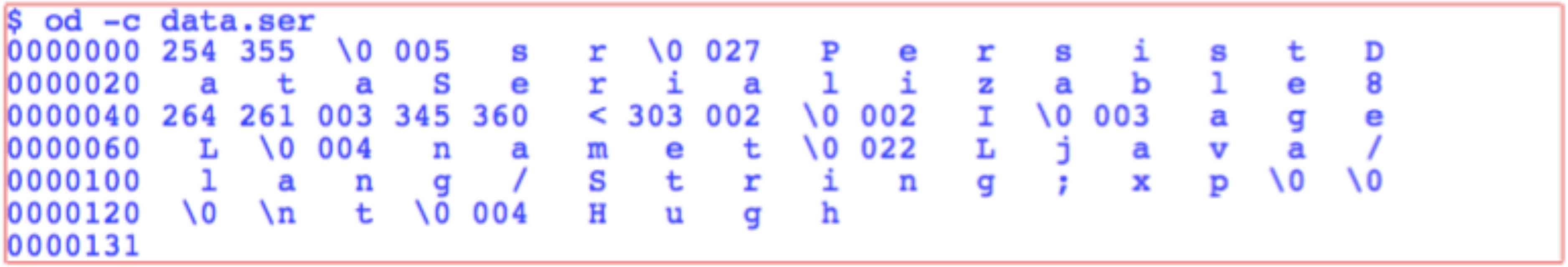
```
Jiu,23,COMP2300,I like cats
Giovanni,39,COMP4300,I like saying, "ciao"
Susan,21,COMP2100,I like computer science
```

Serializable

- Many programming languages have an inbuilt feature for storing objects created by the programming language straight to file
- So for example if we had an object of class “Car” we could save this to file, and then load it back into the program as a “Car” object without any need for writing code to create the specific object.

Example Data Serialized by Java

If we stored an object which held the name and age of a person, in this case Hugh age 10, then Java would store a file like:



```
$ od -c data.ser
0000000 254 355  \0 005   s   r  \0 027   P   e   r   s   i   s   t   D
0000020   a   t   a   S   e   r   i   a   l   i   z   a   b   l   e   8
0000040 264 261 003 345 360   < 303 002  \0 002   I  \0 003   a   g   e
0000060   L  \0 004   n   a   m   e   t  \0 022   L   j   a   v   a   /
0000100   l   a   n   g   /   S   t   r   i   n   g   ;   x   p  \0  \0
0000120  \0  \n   t  \0 004   H   u   g   h
0000131
```

(note that serialized data is saved in binary format)
In terminal we can use od -c datafile to see it in text format

Using Java Serializable

- Declare any class as serializable *“public class SomeClass implements Serializable { }”*
- Doesn’t require implementation of any methods
- Load serialized data by creating an ObjectInputStream object and casting the stream to the appropriate class type
- Save serialized data by creating an ObjectOutputStream and writing the object to the stream

- Arraylists of objects are serializable by default and are a natural choice for serializing a data collection
- Hashmaps are also serializable by default, also many other classes in the standard library

Advantages and Disadvantages

- Easy to implement
- Not necessarily robust across changes in class design
- Security and privacy – serialized objects aren’t secure (issue applies to most other persistence options discussed in these lectures)

Next and References

- Next we'll look xml and json
- References:
  - Kay Horstmann, Big Java (chapter on Files and Streams)
  - IBM developer works 5 things you need to know about serialization https://www.ibm.com/developerworks/library/j-5things1/
  - Oracle serialization faq http://www.oracle.com/technetwork/java/javase/tech/serializationfaq-jsp-136699.html

Australian National University

Persistent Data Overview

Michael Curtotti

Adapted from lectures by

Eric McCreath

Overview

- Persistent Data or Data Storage
- Use cases for persistent data – different kinds of data
- Approaches to Storing and Accessing Persistent Data (bespoke, XML, JSON, serialisation, databases)
- Advantages and Disadvantages of different approaches

Uses of Data and Storage

Type	Use Cases	Formats
Text files (unstructured)	Word processing	Raw text (ascii, utf-8), proprietary word processing formats .doc, (generally unstructured)
Structured text files	Spreadsheet, sensor data, simple structured data (e.g. student data)	csv, tsv, bespoke,
Graphics	Images	png, jpeg (lossy), gif, bmp
Audio/Movie	Lecture recordings, music	mp3, mp4 (lossy)
Data compression	Large file storage	zip, tar

Some common approaches

- Bespoke data file
- XML
- JSON
- Java serialisation
- data bases – e.g. e - Wordpress: php and sql databases

Factors in selecting persistent formats

- Use case – what kind of data/ what does your program do?
- Programming agility – bespoke – no overhead – easy to code – library requirements
- Robustness – bespoke formats don't embed learning of well designed formats such as xml
- Extensibility – can the data format be easily extended to add new fields
- Portability – will other programs need to access the data, run on other hardware?
- Size vs completeness - lossy vs lossless
- audio stream vs financial/scientific data
- Internationalisation – e.g. ascii vs utf-8 (audience)

Databases

- Database management systems commonly used for storage of large volumes of data
- Relational databases – such as mySQL
- Data is stored in large linked tables
- Linking tables through unique identifiers avoids problems of repeating data entry.
- e.g. A Customer table would contain customer data while an invoice table would contain unique identifiers for customers and products
- SQL - designed for data query and manipulation

Next & References

- Next we'll look examples of using bespoke formats and using java's inbuilt serializable features
- References
  - Kay Horstmann, Big Java – relevant chapters (Files and Streams, Relational Databases, XML)

Australian National University

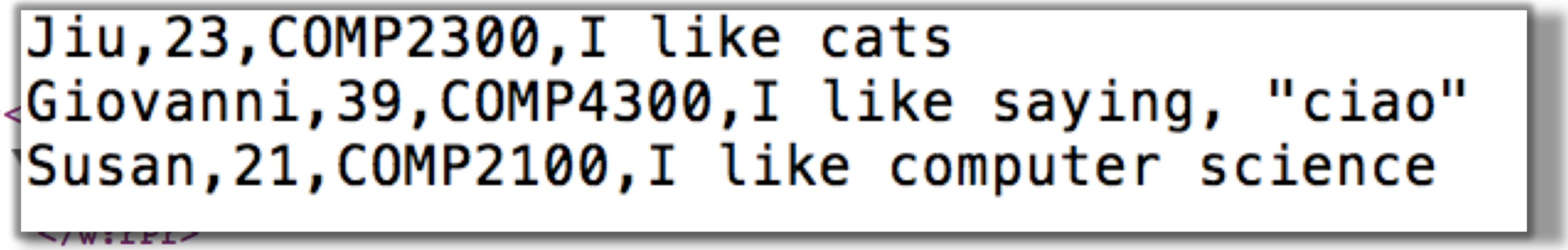
Persistent Data XML and JSON

Michael Curtotti
Adapted from lectures by
Eric McCreath

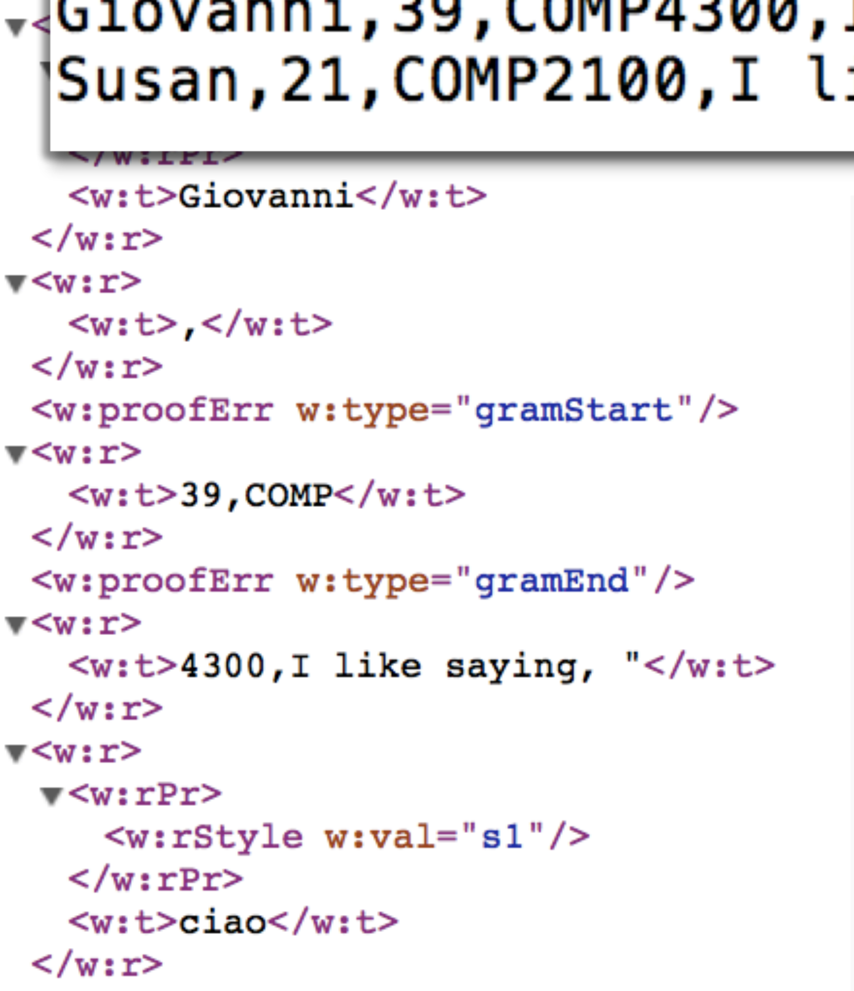
Overview

- Explore XML and JSON as approaches to data persistence
- Demonstrate the approaches by looking at simple implementations

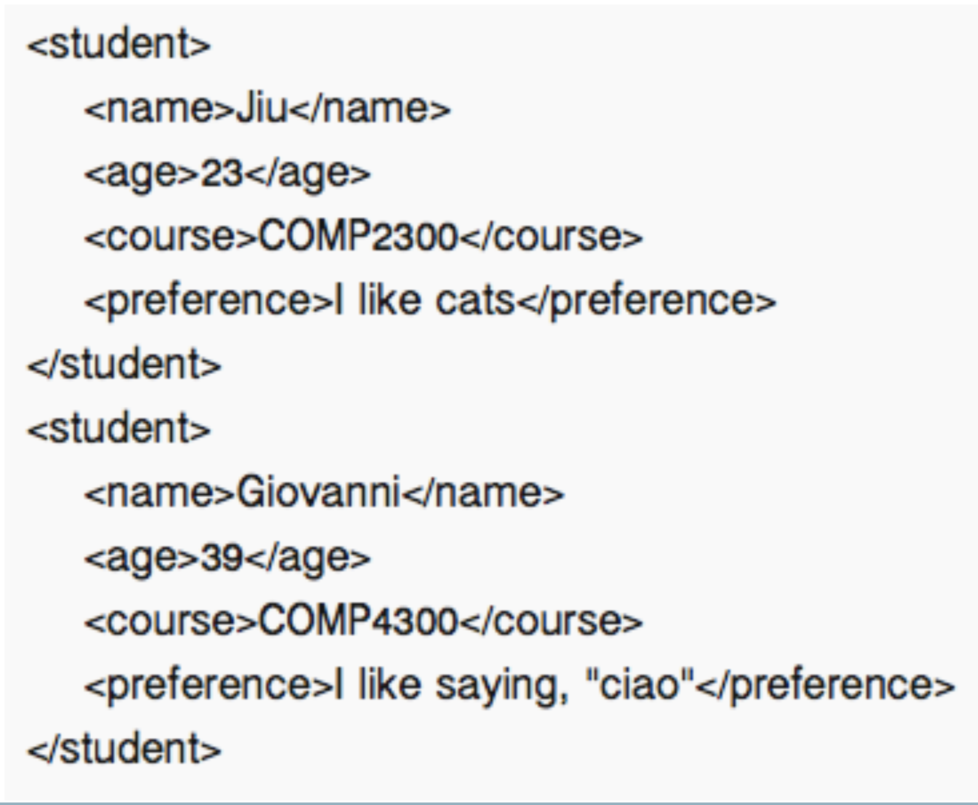
XML - Examples



```
Jiu,23,COMP2300,I like cats
Giovanni,39,COMP4300,I like saying, "ciao"
Susan,21,COMP2100,I like computer science
```



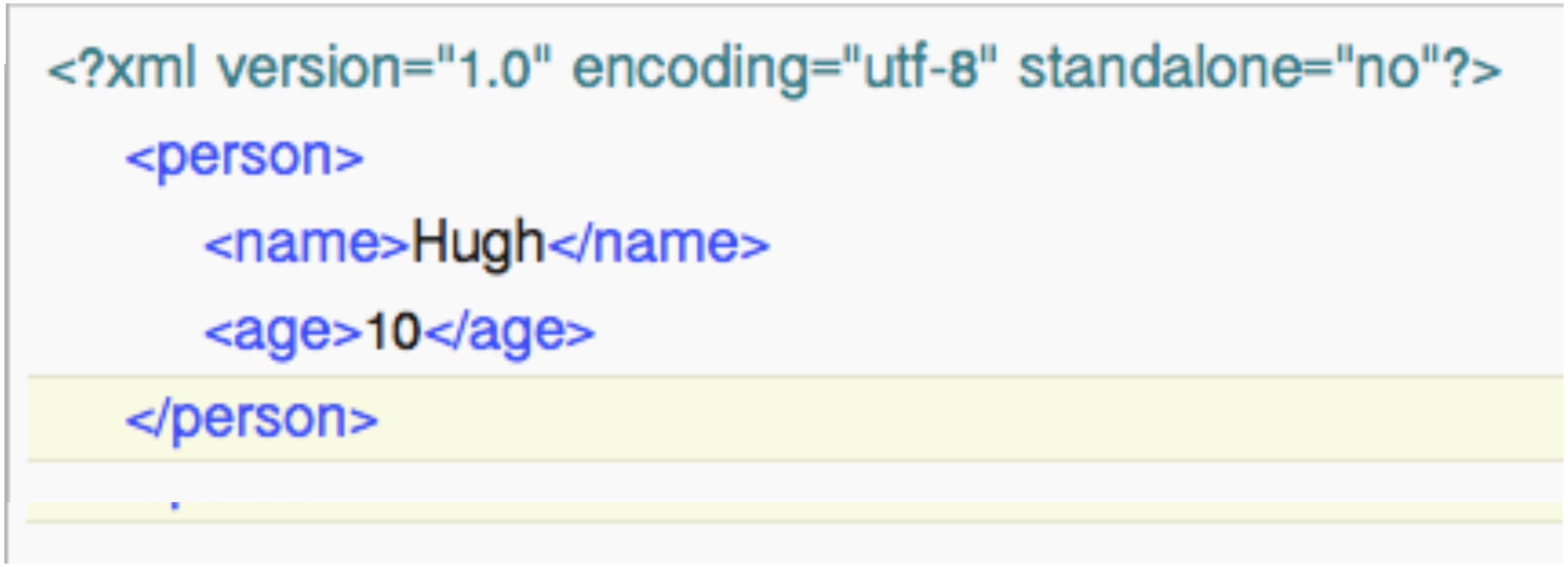
```
▼<
  </w:rPr>
    <w:t>Giovanni</w:t>
  </w:r>
▼<w:r>
    <w:t>,</w:t>
  </w:r>
  <w:proofErr w:type="gramStart"/>
▼<w:r>
    <w:t>39,COMP</w:t>
  </w:r>
  <w:proofErr w:type="gramEnd"/>
▼<w:r>
    <w:t>4300,I like saying, "</w:t>
  </w:r>
▼<w:r>
  ▼<w:rPr>
      <w:rStyle w:val="s1"/>
    </w:rPr>
    <w:t>ciao</w:t>
  </w:r>
```



```
<student>
    <name>Jiu</name>
    <age>23</age>
    <course>COMP2300</course>
    <preference>I like cats</preference>
</student>
<student>
    <name>Giovanni</name>
    <age>39</age>
    <course>COMP4300</course>
    <preference>I like saying, "ciao"</preference>
</student>
```

Australian National University

Implementation Example



```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
  <person>
    <name>Hugh</name>
    <age>10</age>
  </person>
```

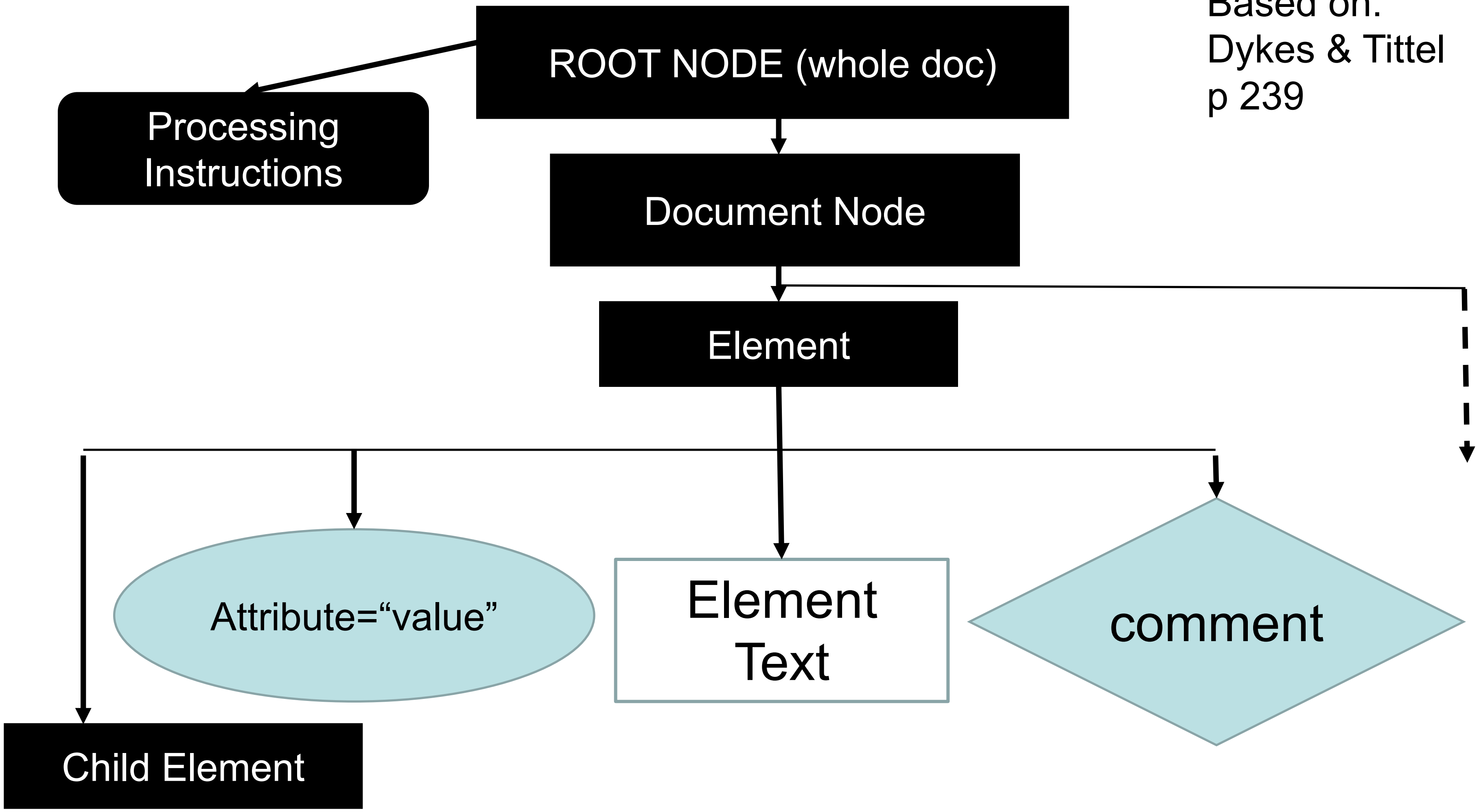
Attribute example <person gender=“male” status=“enrolled”>

Two major options for XML in Java

- Two approaches – DOM and SAX
- *SAX – Simple API for XML*
- *DOM – Document Object Model (structured around XML standard)*
- SAX treats xml as stream and allows extraction of data as stream is read – preferable for very large documents (gigabyte)
- SAX very fast and efficient as compared to DOM
- Java DOM reads in entire XML tree and generates a tree object

XML - DOM

Based on:
Dykes & Tittel
p 239



Using DOM

- DOM requires a number of steps to save data to file:
  - DocumentBuilderFactory used to generate a DocumentBuilder
  - Document created from a DocumentBuilder object
  - Then a matter of creating appropriate elements and appending them in appropriate tree –
  - eg. age element a child of person element -append text node as age element
  - Then further steps to output to file
- DOM load similar series of steps
  - DocumentBuilderFactory
  - Document Builder
  - Document
  - Class data structures

Advantages and Disadvantages

- Robust
- Portable
- Extensible
- Involved to implement
- Human readable – a positive but also an issue for security and privacy

Australian National University

JSON

- Javascript Object Notation (JSON), like XML, is also an open standard format that is widely used.
- Originally designed for sending data between web client and server, but very useful for data storage also.
- Built around attribute-value pairs and produces smaller and more readable documents than XML.
- e.g in JSON: {"age":10,"name":"Hugh"}

Advantages and Disadvantages

- There is no inbuilt Java library for JSON, however, there are a number of open-source libraries available (our lab machines have json-simple.jar installed).
- Similar advantages/disadvantages to XML however
  - more lightweight
  - straightforward to implement
  - Not native to standard Java library
  - lacking language features of XML

JSON syntax and types

{“attribute-name”:{JSON object}}

{“attribute-name”:”string”}

{“attribute-name”:[array]}

{“attribute-name”:number}

{“attribute-name”:boolean}

{“attribute-name”:null}

Example Exam questions

- What are some advantages in storing information in either XML and JSON compared to sorting information using the Serializable approach?
- List some differences and similarities between XML and JSON.
- Modify the code **PersistDataXML.java** such that it can load and save a list of people rather than just a single person as it currently does.
- Persistence code available in the git repository

References

- References:
  - Kay Horstmann, Big Java (chapter on XML)
  - Elliotte Rusty Harold, Processing XML with Java, Addison-Wesley 2006
  - Lucinda Dykes & Ed Tittel, XML for Dummies, Wiley Publishing 2005
  - W3C XML standards pages http://www.w3.org/XML/
  - JSON http://www.json.org
  - The JSON Data Interchange Format http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
  - 10 Examples of JSON Files https://www.sitepoint.com/10-example-json-files/