Week 8 Announcements

- Video Assignment #2 → No late submission (hard deadline, Friday, 27 October) | 6 weeks.
- Escape Room will happen this week!
- Group Project:
  - Mystery: it's been over two weeks! Have you solved it?
  - Have you checked the content about general tips?
  - Checkpoint 1: this week!
  - Checkpoint 2: Week 10.
- This Friday, 29 September, we'll have an additional lecture:
  - Persistent Data
- Final exam: it will be in-person in the computing labs.
  - It's highly recommended to attend at least once to the labs to experience the ambience.

Australian National University

COMP2100/6442
Software Design Methodologies / Software Construction

Persistent Data

Bernardo Pereira Nunes and
Sergio J. Rodríguez Méndez

Outline

> Persistent Data

> Bespoken

> Serialisation

> XML

> JSON

>> Pros and Cons

What is Persistent Data?

A critical task for applications (save/restore data)

> Permanent data (storage of data from working memory)

> It can be updated, but not as frequent as transient/volatile data

> It is stored in database/SSD/HD/Magnetic tape

> Why do we want permanent data?

> Easy! Think about your bank account as volatile data.

> Logging information

> To be used and reused (save and load)

> How can we persist data?

> The choice of the persistence method is part of the design of an application

> Files (JSON, XML, images, …)

> Databases

Uses of Data and Storage

Types	Use cases	Formats
Text files (unstructured data)	Word Processing	raw text (ASCII, UTF-8) proprietary word processing formats .doc (generally unstructured)
Structured text files	Spreadsheet, sensor data, simple structured 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, rar, …

How to determine which is the best format to persist your data?

> Use case

> What does your application do?

> What kind of data you have?

> Is there any restriction to meet (license, storage limitation, rapid access to data, rapid development,…)?

Aspects to consider

> Programming Agility

>> Easy to develop (no overhead) and code

> Extensibility

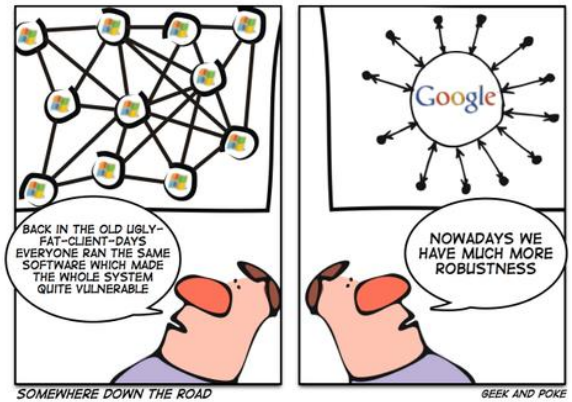
>> Can data be easily extended? (e.g., add new fields, attributes, …)

>> Is it easy to add new fields in a CSV file?
Is it easy to add new attributes in a graph database?

> Portability

>> Important! Will **other applications access the data**? Will it run on **other hardware**?

Aspects to consider



> Robustness

>> Bespoke vs XML vs JSON

>>> well-designed and structured format

>>> No schema (how verify if your data is correctly formatted?)

>>> Lack of schema > interoperability problems

> Size vs Completeness

>> Lossy vs Lossless

>>> Audio/Image vs financial data / scientific data

> Internationalisation

>> ASCII vs UTF-8

>> Who will use the data (audience)?

Databases

PostgreSQL ORACLE DATABASE Microsoft SQL Server MySQL MariaDB

> Database management systems (DBMS) are commonly used for storage of **large volumes of data**

> **Relational databases** – (e.g., MySQL)

>> Linking tables through unique identifiers avoids problems of repeating data entry

>> Example (next slide)

Databases - Example

Represent a person in a bespoke/csv file:

id, fullname, homePhone, mobilePhone, workPhone

1, Bernardo, 1234, 3210, 9898

2, John, ? ,1230, ?

Relational Database (RDB)

SQL (Structure Query Language) designed for data query and manipulation

Person

id	fullname
1	Bernardo
2	John
...	…

0..n

1..1

ContactPhones

personId	phoneNumber
1	555-555
2	123-222
1	123-321

(not that this representation is oversimplified)

Demonstration

> Bespoke

> Implement a simple logging application

> Save/load log errors to/from a text file

listoferrors.txt

```
101, 2020-07-16T08:44:23.802853, IOException occurred, severe, Bespoke, loadData
102, 2020-07-16T08:44:23.802853, Exception occurred, low, Bespoke, saveData
```

Demonstration

> Java Serialisation

> Implement a simple application

> Terminal command:

```
od -c data.ser || PS_> Format-Hex data.ser
```

```
0000000  254 355  \0 005    s    r  \0 017    P    D    S    e    r    i    a    l
0000020    i    z    a    t    i    o    n 370 314    u 033 322 303 024    F 002
0000040   \0 002    I  \0 002    i    d    L  \0 004    n    a    m    e    t   \0
0000060  022    L    j    a    v    a    /    l    a    n    g    /    S    t    r    i
0000100    n    g    ;    x    p   \0   \0   \0 031    t   \0   \b    B    e    r    n
0000120    a    r    d    o 254 355  \0 005    s    r  \0 017    P    D    S    e
0000140    r    i    a    l    i    z    a    t    i    o    n 370 314    u 033 303
0000160  303 024    F 002   \0 002    I  \0 002    i    d    L  \0 004    n    a
0000200    m    e    t   \0 022    L    j    a    v    a    /    l    a    n    g    /
0000220    S    t    r    i    n    g    ;    x    p   \0   \0   \0 031    t   \0   \b
0000240    B    e    r    n    a    r    d    o
```

Bespoken and Serialisation

> Not very used in industry

>> Not robust enough

> **Serialisation** presents technical issues such as:

>> It may **depend on the programming language**

>> Lose object references

>> **Security issues**

>> Endianness (big- or little-endian)

Demonstration

> Java Serialisation

>> Class must implement Serializable

>>> *public myClass implements Serializable*

> Load serialisable data by creating an **ObjectInputStream** object and casting the stream to the appropriate class type

> Save serialised data by creating an **ObjectOutputStream** and writing the object to the stream

> ArrayLists are serialisable by default and are commonly used for serialising data collections (many classes, such as HashMaps, are serialisable (check documentation))

Demonstration

Warning! Deserialisation of untrusted data is inherently dangerous and should be avoided !

8 Serialization and Deserialization

Note: Deserialization of untrusted data is inherently dangerous and should be avoided.

Java Serialization provides an interface to classes that sidesteps the field access control mechanisms of the Java language. /
Furthermore, deserialization of untrusted data should be avoided whenever possible, and should be performed carefully wh

Guideline 8-1 / SERIAL-1: Avoid serialization for security-sensitive classes

Security-sensitive classes that are not serializable will not have the problems detailed in this section. Making a class serializa
adds a hidden public constructor to a class, which needs to be considered when trying to restrict object construction.

Similarly, lambdas should be scrutinized before being made serializable. Functional interfaces should not be made serializab

Guideline 8-2 / SERIAL-2: Guard sensitive data during serialization

!

Once an object has been serialized the Java language's access controls can no longer be enforced and attackers can access
sensitive data in a serializable class.

https://www.oracle.com/java/technologies/javase/seccodeguide.html

Demonstration

> XML

> Implement simple example

> Save/load to/from text file

> .docx (Word document) is represented using XML

Demonstration

XML Structure / Tree

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<root>
  <child attributes="0">
    <subchild>…</subchild>
    <subchild>…</subchild>
     …
  </child>
  <child>
    <subchild>…</subchild>
    <subchild>…</subchild>
  </child>
…
</root>
```

XML is case sensitive! <Root> != <root>

Demonstration

XML parser error! “<” you must use <

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<root>
  <child attributes=“0”>
    <subchild> 10 < x < 100 </subchild>
    <subchild>…</subchild>
     …
  </child>
  <child>
    <subchild>…</subchild>
    <subchild>…</subchild>
  </child>
…
</root>
```

There are 5 pre-defined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

*Only < and & are strictly illegal in XML, but it is a good habit to replace > with > as well.

*https://www.w3schools.com/xml/xml_syntax.asp

Demonstration

XML Example

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<People>
    <Person id= "1">
        <FirstName>Johnny</FirstName>
        <LastName>B. Good</LastName>
    </Person>
    <Person id= "2">
        <FirstName>Bart</FirstName>
        <LastName>Simpson</LastName>
    </Person>
</People>
```

Two major options for XML in Java

> Two approaches: **DOM** and **SAX**

> SAX – Simple API for XML

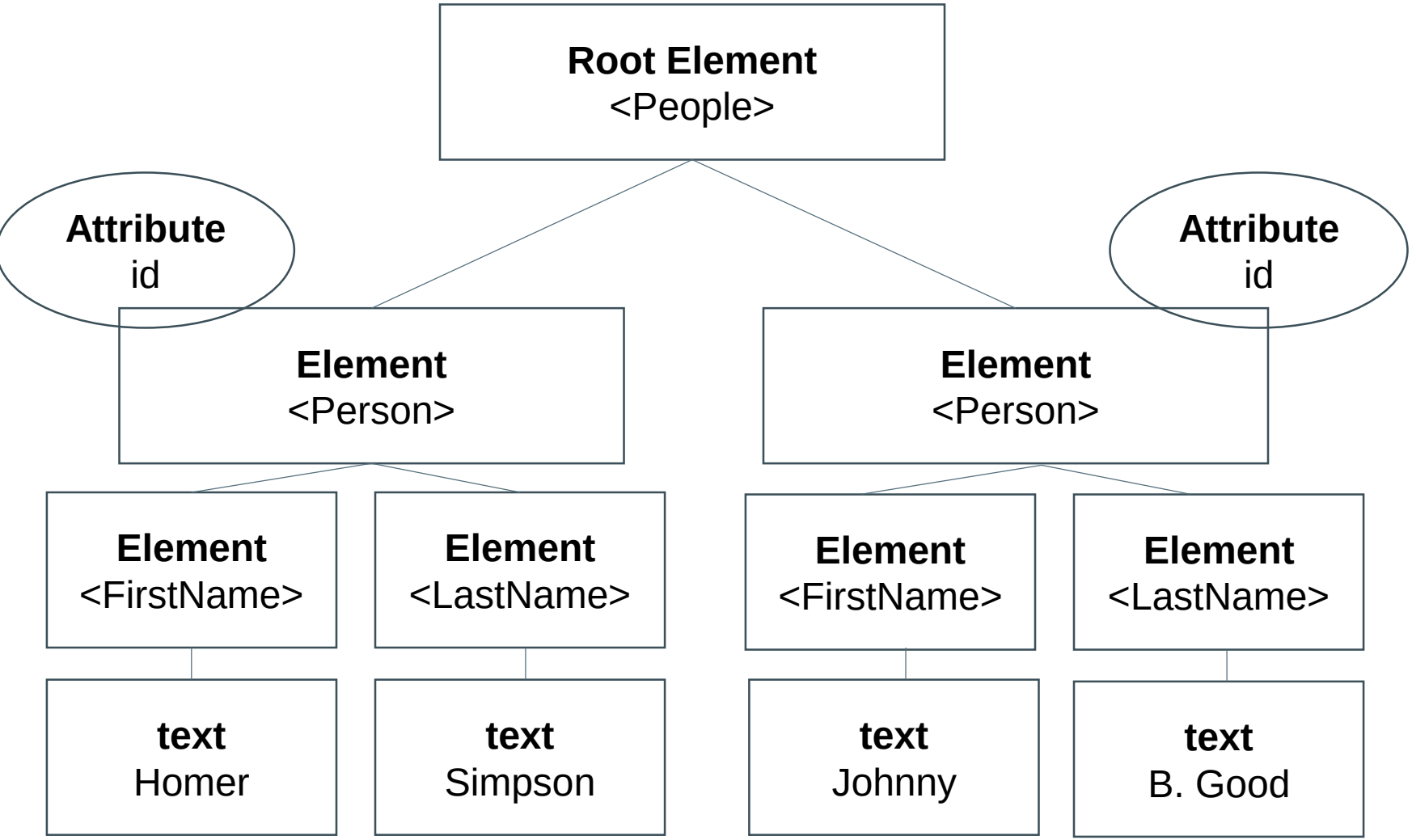
> DOM – Document Object Model (structured around XML standard)

> SAX treats XML as stream and allows extraction of data as stream is read - preferable for very large documents (gigabyte)

> SAX is very fast and efficient compared to DOM

> Java DOM reads the entire XML tree and generates the node object

XML DOM


Root Element
<People>
Attribute
id
Attribute
id
Element
<Person>
Element
<Person>
Element
<FirstName>
Element
<LastName>
Element
<FirstName>
Element
<LastName>
text
Homer
text
Simpson
text
Johnny
text
B. Good

XML DOM

DOM requires a number of steps to save data to file:

- Create a DocumentBuilder (uses DocumentBuilderFactory)
- Document created from a DocumentBuilder object
- Create and append elements
- Transform the XML to a Result (output file)

Similar series of steps for loading XML/DOM:

- DocumentBuilderFactory
- Document Builder
- Document
- Class data structures

Some Related Packages

```
import javax.xml.parsers.*;
import javax.xml.transform.*;
import org.w3c.dom.*;
```

XML — Complementary Resources

Fundamentals and Standards Family:

- W3C Standards Overview (by Sergio Rodríguez Méndez).pdf
- XSLT + XPath (demo):
  https://www.w3schools.com/xml/xsl_for_each.asp
  https://www.w3schools.com/xml/xsl_choose.asp

Applications:

- MS Office (PPTX, DOCX, etc.) (demo).
- MathML (demo).
- SVG (demo).

XQuery:

https://www.w3.org/TR/xquery-31/
http://www.xqueryfunctions.com/
https://www.saxonica.com/welcome/welcome.xml
https://www.videlibri.de/cgi-bin/xidelcgi

Pros and Cons

+ Robust, Extendable
+ Human readable
+ Portable, Platform independent and programming language independent
+ XML supports Unicode (international encoding)
+ Easy format verification
+ it can represent data structures (trees, lists…)

- XML syntax is verbose and redundant
- XML file sizes are usually big because of above
- Does not support Array

JSON

https://www.json.org/

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 also very useful for data storage.

Built around attribute-value pairs and produces smaller and more readable documents than XML.

JSON example: [{"age":11,"name":"Bart"},{"age":40,"name":"Homer"}]

http://json.parser.online.fr

JSON

```
{“attribute-name”:{JSON object}}
{“attribute-name”:“string”}
{“attribute-name”:[array]}
{“attribute-name”:1} (number)
{“attribute-name”:true} (boolean)
{“attribute-name”:null}
```

JSON Pros and Cons

+ More lightweight
+ Straightforward to implement
+ Support array and null
+ It can easily distinguish boolean, number, and string type
+ Data is available as JSON objects

- Lacking language features of XML (e.g., XML attributes..)
- No native support in Java (XML is 100% compatible)
- It has no display capabilities (no markup language)

Exercise

Which alternative is INCORRECT about persistent data:

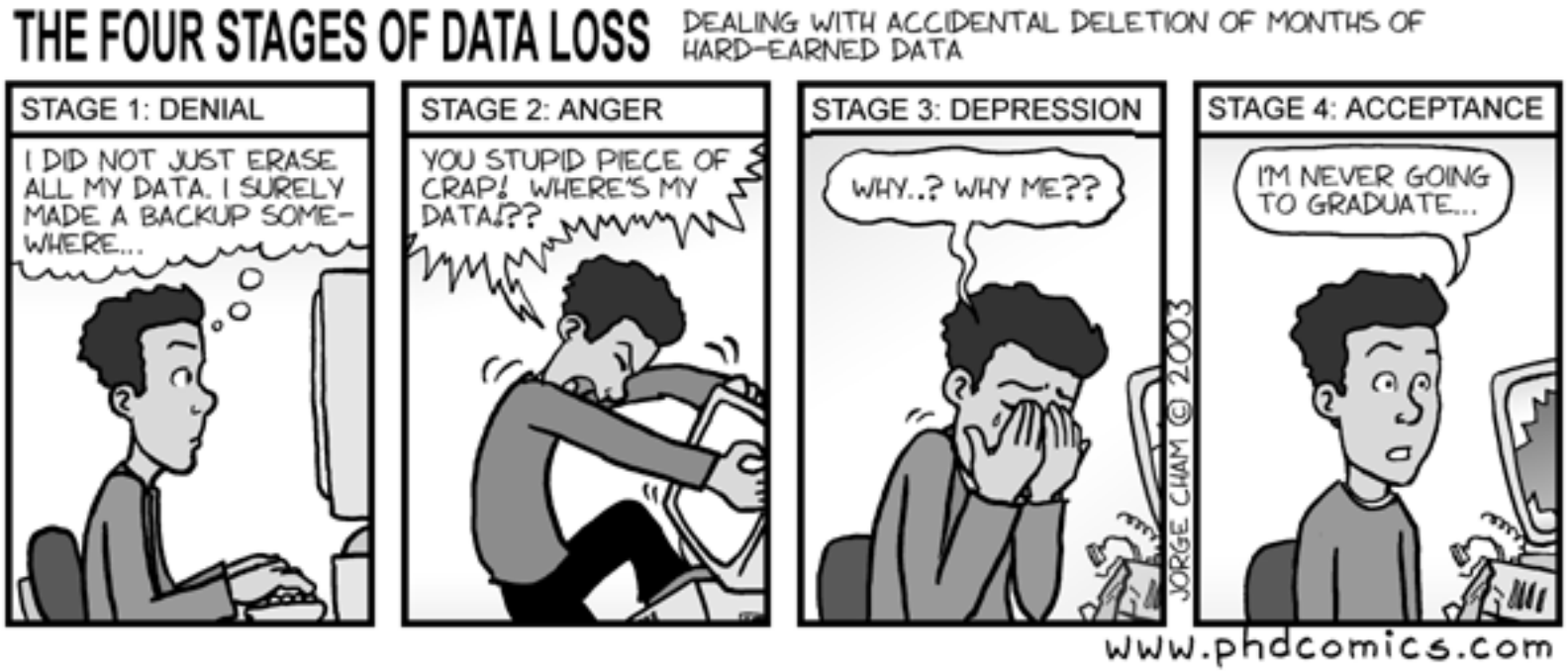
a) JSON supports array and null values.
b) XML is robust, extensible, and human-readable. Also, JSON tends to be more lightweight than XML.
c) XML is case insensitive. It means that <root> is equal to <ROOT>.
d) There is no native support in Java to handle JSON files.

Exercise

Which one of the following data formats can not be used across multiple programming languages?

a) XML
b) Object serialisation
c) JSON
d) CSV file

Meme for today's lecture!



Recommended Reading

> Kay Horstmann, Big Java (Chapters on Files and Streams, Relational Databases, XML)

> IBM developer works 5 things you need to know about serialisation

https://developer.ibm.com/technologies/java/articles/j-5things1/

> Oracle serialization FAQ

https://www.oracle.com/technetwork/java/javase/tech/serializationfaq-jsp-136699.html

Recommended Reading

> W3C XML standards pages https://www.w3.org/standards/

> JSON https://www.json.org/

> The JavaScript Object Notation (JSON) Data Interchange Format https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

> 10 JSON Examples to Use in Your Projects https://www.sitepoint.com/10-example-json-files/