Authoring Efficient XML

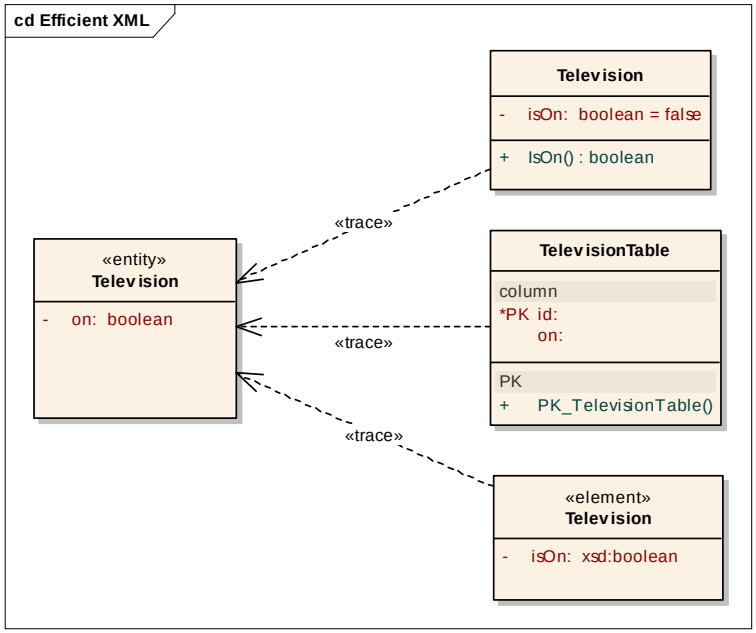
A guide to writing XML that is efficient to transmit and parse

What is XML?

- A *specification* from the W3C
- A *meta-language*: A language for expressing **languages** that can express data structures
- A *text based*, human-readable markup
- A *derivative* of the SGML
- A set of rules: “Well formed XML”

Representing Entities

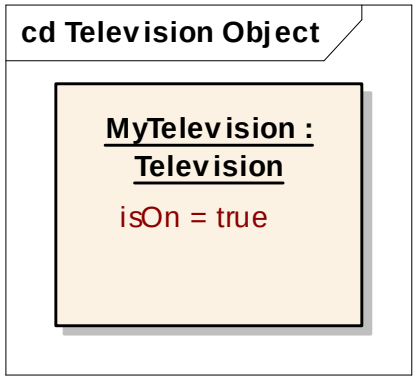
- XML *elements* are an expression of *entities* in a domain, just classes or relational database tables
- Classes encapsulate *behaviour* **and** *data structure*
- Elements and tables encapsulate only *data structure*



Representing Entities

Types	Objects
XML Element Types	XML Elements
Tables	Table Rows
Classes	Objects
Logical Entity Types	Real Life Objects

<<define>>



are equivalent

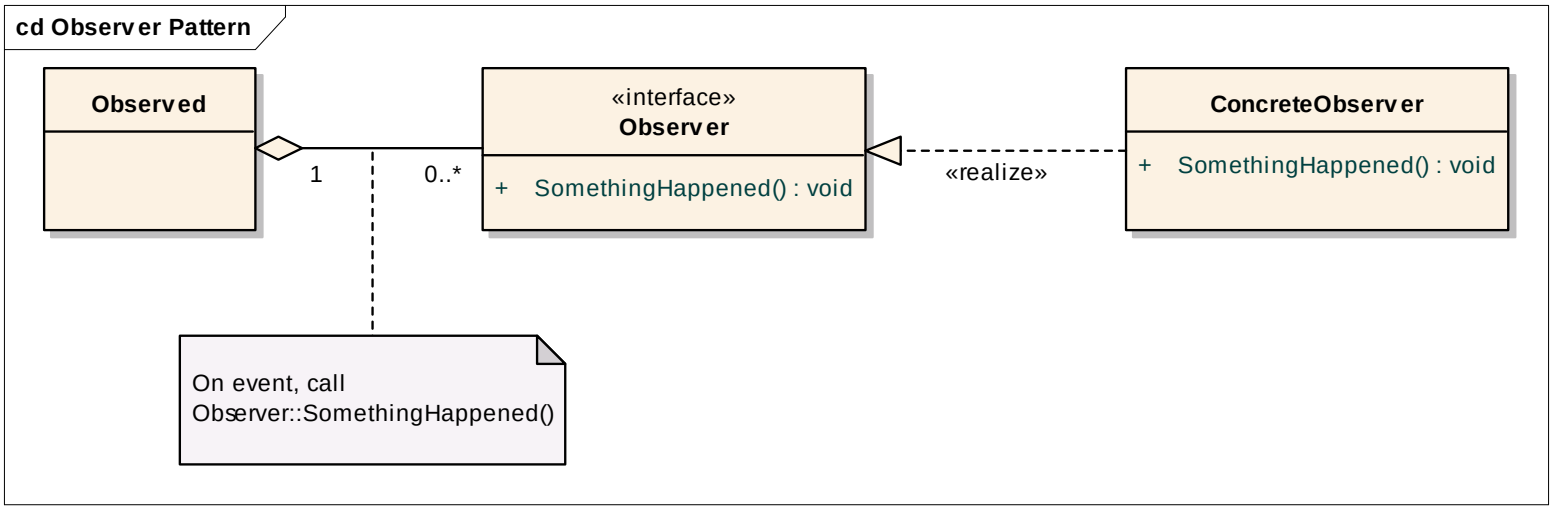
```
<Television isOn="true" />
```

Efficiency Objectives

1. To minimise the physical size of the XML to optimise transmission rates
   - How can we represent a given data structure using a *minimum* number of characters?
2. To minimise the time taken for parsers to parse XML
   - How can we structure our XML to minimise the amount of time it takes to parse it?

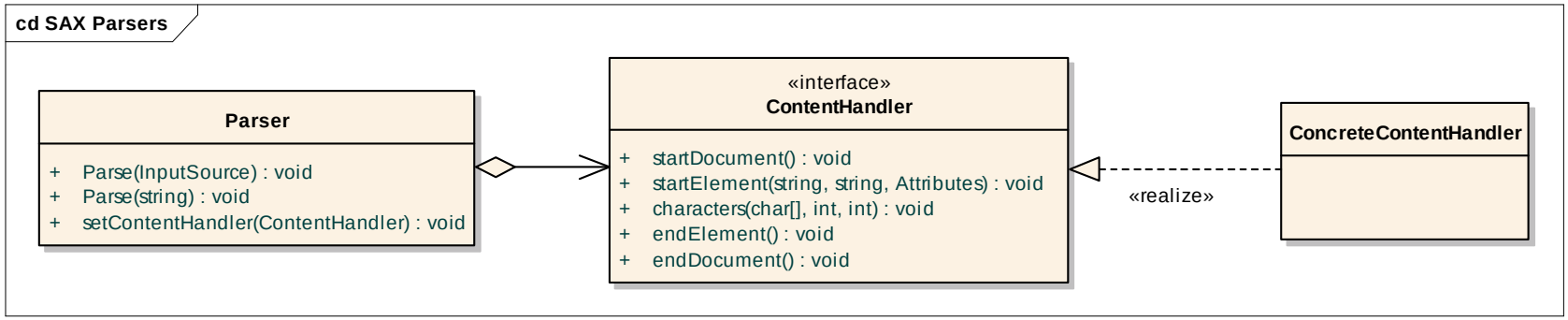
Simple XML Parsers

- "Simple API for XML" (SAX) parsers implement the *observer pattern*
- Listeners (Java) and events (.NET) are other implementations of *observers*



Simple XML Parsers

- The *Parser* raises *events* about the document, elements, attributes and text when it *finds* them in the XML source
- These are caught by a *ContentHandler*, which knows how to deal with them (e.g. constructs a XML DOM)
- We can help improve parse efficiency by structuring our XML to *minimise* the number of SAX events raised



Simple XML Parsers

- Other XML parsers work in different ways
- E.g. the Microsoft .NET Xml library

Rules of Thumb

Rule 1: Attributes not Elements when size is important

- Use attributes to express *named* data items rather than elements
- This uses roughly ½ the number of text characters…

```
<Television isOn=“true” />
```

```
<Television>
        <isOn>true</isOn>
</Television>
```

Rule 2: Elements not Attributes when parse time is important

- …and can **reduce** parse time by **reducing** the number of SAX event calls invoked
- **But**, there is a trade-off with additional time taken to build *Attributes* passed in the `startElement()` SAX event!
- On average, better parse times are achieved using Attributes for smaller data sets (< 300K bytes) and use Elements for larger ones

```
<Television isOn="true" />
```

```
<Television>
        <isOn>true</isOn>
</Television>
```

Rule 2: Elements not Attributes when parse time is important

- Other XML parsers vary in terms of parse performance
- Microsoft .NET Xml libraries generally perform better with data in Elements rather than Attributes

```
<Television isOn="true" />
```

```
<Television>
        <isOn>true</isOn>
</Television>
```

Rule 3: Don’t use “envelopes”

- Don’t enclose a group of elements of the same name inside an “envelope”
- This expresses explicitly an implicit relationship!
- Envelopes demand text and parse effort unnecessarily

```
<Television>
        <Buttons>
                <Button depressed=“true” />
                <Button depressed=“false” />
        </Buttons>
</Television>
```

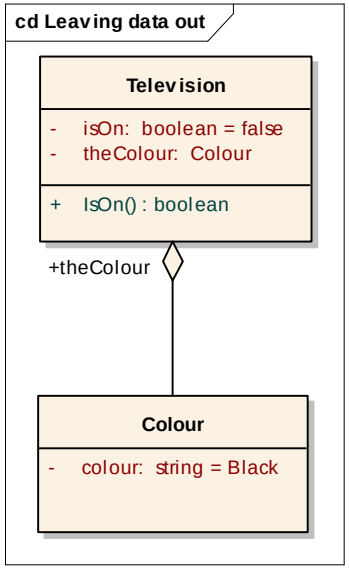
Rule 3: Don’t use “envelopes”

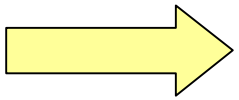
- You may need to *logically group* a set of elements of the same name within a *group element* to represent that they are related
- This is **not enveloping**: The *group element* represents a logical entity in the system

```
<Television>
        <ButtonGroup>
                <Button depressed=“true” />
                <Button depressed=“false” />
        </ButtonGroup>
        …
</Television>
```

Rule 4: De-normalize the data

- Normalize the data structure according to purpose
- Consider what can be left out in terms of relationships and data!





```
<Television isOn="" colour="" />
```

Rule 5: Avoid unnecessary white space

- Avoid using white space when assembling production XML
- Computers don’t care whether a human can read the XML or not!

```
<Television isOn=“true”><Screen size=“24”/></Television>
```

```
<Television isOn=“true”>
        <Screen size=“24” />
</Television>
```

Rule 6: Use short-hand techniques

- Use short versions of XML syntax where possible
- This simply reduces characters used!

```
<Television isOn=“true” />
```

```
<Television isOn=“true”></Television>
```

Rules Summary

1. Favour **attributes** over elements where size is important
2. Favour **elements** over attributes where parse time is important
3. Don’t use **envelopes**
4. De-**normalize** where possible
5. Avoid unnecessary **white space**
6. Use **short-hand** techniques