

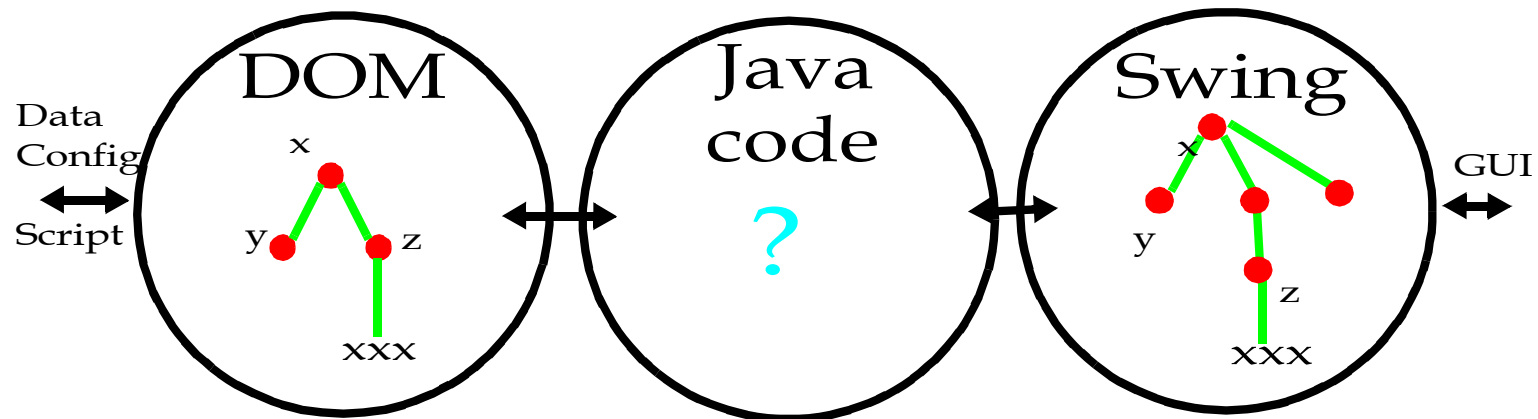
### 3) XML and Java

- *XML gives Java something to do* (Jon Bosak, Sun)
- *XML is fundamental to our plans for the next generation enterprise-computing platform* (Bill Roth, Sun)
- *Combining Java and XML technologies produces portable 'smart' data* (Anne Thomas, Patricia Seybold Group)
- *XML technology makes the information exchange possible, and Java technology makes automation feasible* (Bill Smith, Sun)

# Design

DOM=Model

SWING=View



# SWING: View

- Java Foundation Class (JFC)
- `java.swing.text` package
- Sample of these interfaces:

*Document* Container for text that supports editing and provides notification of changes (serves as the model in an MVC relationship).

*StyledDocument* Interface for a generic styled document.

*Element* Interface to describe a structural piece of a document.

*AttributeSet* A collection of unique attributes.

*AttributeSet.CharacterAttribute* This interface is the type signature that is expected to be present on any attribute key that contributes to character level presentation.

*AttributeSet.ColorAttribute* This interface is the type signature that is expected to be present on any attribute key that contributes to presentation of color.

*AttributeSet.FontAttribute* This interface is the type signature that is expected to be present on any attribute key that contributes to the determination of what font to use to render some text.

*AttributeSet.ParagraphAttribute* This interface is the type signature that is expected to be present on any attribute key that contributes to the paragraph level presentation.

*Style* A collection of attributes to associate with an element in a document.

# DOM: Document Object Model

## Java API

```
public interface DOMImplementation {
    public boolean        hasFeature(String feature,
                                     String version);
}

public interface Node {
    // NodeType
    public static final short    ELEMENT_NODE           = 1;
    public static final short    ATTRIBUTE_NODE         = 2;
    public static final short    TEXT_NODE              = 3;
    public static final short    CDATA_SECTION_NODE     = 4;
    public static final short    ENTITY_REFERENCE_NODE  = 5;
    public static final short    ENTITY_NODE            = 6;
    public static final short    PROCESSING_INSTRUCTION_NODE = 7;
    public static final short    COMMENT_NODE           = 8;
    public static final short    DOCUMENT_NODE          = 9;
    public static final short    DOCUMENT_TYPE_NODE    = 10;
    public static final short    DOCUMENT_FRAGMENT_NODE = 11;
}
```

```

public static final short          NOTATION_NODE          = 12;

public String                     getNodeName();
public String                     getNodeValue()
                                throws DOMException;
public void                       setNodeValue(String nodeValue)
                                throws DOMException;

public short                      getNodeType();
public Node                      getParentNode();
public NodeList                  getChildNodes();
public Node                      getFirstChild();
public Node                      getLastChild();
public Node                      getPreviousSibling();
public Node                      getNextSibling();
public NamedNodeMap              getAttributes();
public Document                  getOwnerDocument();
public Node                      insertBefore(Node newChild,
                                Node refChild)
                                throws DOMException;
public Node                      replaceChild(Node newChild,
                                Node oldChild)
                                throws DOMException;
public Node                      removeChild(Node oldChild)
                                throws DOMException;
public Node                      appendChild(Node newChild)
                                throws DOMException;
public boolean                   hasChildNodes();
public Node                      cloneNode(boolean deep);
}

```

```
public interface NodeList {
    public Node        item(int index);
    public int         getLength();
}

public interface NamedNodeMap {
    public Node        getNamedItem(String name);
    public Node        setNamedItem(Node arg)
                        throws DOMException;
    public Node        removeNamedItem(String name)
                        throws DOMException;
    public Node        item(int index);
    public int         getLength();
}

public interface DocumentFragment extends Node {
}
```

```

public interface Document extends Node {
    public DocumentType      getDoctype();
    public DOMImplementation getImplementation();
    public Element           getDocumentElement();
    public Element           createElement(String tagName)
                               throws DOMException;

    public DocumentFragment createDocumentFragment();
    public Text              createTextNode(String data);
    public Comment           createComment(String data);
    public CDATASection      createCDATASection(String data)
                               throws DOMException;

    public ProcessingInstruction createProcessingInstruction(String
target,
                                                             String data)
                                                             throws
DOMException;
    public Attr              createAttribute(String name)
                               throws DOMException;
    public EntityReference   createEntityReference(String name)
                               throws DOMException;
    public NodeList          getElementsByTagName(String tagname);
}

```

```

public interface Element extends Node {
    public String      getTagName();
    public String      getAttribute(String name);
    public void        setAttribute(String name,
                                   String value)
                                   throws DOMException;
    public void        removeAttribute(String name)
                                   throws DOMException;
    public Attr        getAttributeNode(String name);
    public Attr        setAttributeNode(Attr newAttr)
                                   throws DOMException;
    public Attr        removeAttributeNode(Attr oldAttr)
                                   throws DOMException;
    public NodeList    getElementsByTagName(String name);
    public void        normalize();
}

```

```

public interface Attr extends Node {
    public String      getName();
    public boolean     getSpecified();
    public String      getValue();
    public void        setValue(String value);
}

```

```

public interface CharacterData extends Node {
    public String      getData()
                        throws DOMException;
    public void        setData(String data)
                        throws DOMException;
    public int         getLength();
    public String      substringData(int offset,
                                     int count)
                        throws DOMException;
    public void        appendData(String arg)
                        throws DOMException;
    public void        insertData(int offset,
                                  String arg)
                        throws DOMException;
    public void        deleteData(int offset,
                                   int count)
                        throws DOMException;
    public void        replaceData(int offset,
                                   int count,
                                   String arg)
                        throws DOMException;
}

public interface Text extends CharacterData {
    public Text        splitText(int offset)
                        throws DOMException;
}

public interface Comment extends CharacterData {
}

public interface CDATASection extends Text {
}

```

```

public interface DocumentType extends Node {
    public String          getName();
    public NamedNodeMap   getEntities();
    public NamedNodeMap   getNotations();
}

public interface Notation extends Node {
    public String          getPublicId();
    public String          getSystemId();
}

public interface Entity extends Node {
    public String          getPublicId();
    public String          getSystemId();
    public String          getNotationName();
}

public interface EntityReference extends Node {
}

public interface ProcessingInstruction extends Node {
    public String          getTarget();
    public String          getData();
    public void           setData(String data)
                          throws DOMException;
}

```

```

public abstract class DOMException extends RuntimeException {
    public DOMException(short code, String message) {
        super(message);
        this.code = code;
    }
    public short    code;
    // ExceptionCode
    public static final short    INDEX_SIZE_ERR           = 1;
    public static final short    DOMSTRING_SIZE_ERR       = 2;
    public static final short    HIERARCHY_REQUEST_ERR    = 3;
    public static final short    WRONG_DOCUMENT_ERR       = 4;
    public static final short    INVALID_CHARACTER_ERR    = 5;
    public static final short    NO_DATA_ALLOWED_ERR      = 6;
    public static final short    NO_MODIFICATION_ALLOWED_ERR = 7;
    public static final short    NOT_FOUND_ERR            = 8;
    public static final short    NOT_SUPPORTED_ERR        = 9;
    public static final short    INUSE_ATTRIBUTE_ERR      = 10;
}

// ExceptionCode
public static final short    INDEX_SIZE_ERR           = 1;
public static final short    DOMSTRING_SIZE_ERR       = 2;
public static final short    HIERARCHY_REQUEST_ERR    = 3;
public static final short    WRONG_DOCUMENT_ERR       = 4;
public static final short    INVALID_CHARACTER_ERR    = 5;
public static final short    NO_DATA_ALLOWED_ERR      = 6;
public static final short    NO_MODIFICATION_ALLOWED_ERR = 7;
public static final short    NOT_FOUND_ERR            = 8;
public static final short    NOT_SUPPORTED_ERR        = 9;
public static final short    INUSE_ATTRIBUTE_ERR      = 10;
}

```

