18A

Android
Reading XML Data
Using SAX and W3C Parsers

Victor Matos

Cleveland State University

Notes are based on:

Android Developers http://developer.android.com/index.html

XML Data http://www.w3.org

http://www.saxproject.org/

XML Data

What is XML?

- **Extensible Markup Language** (**XML**) is a set of rules for encoding documents in a readable form.
- Similar to HTML but <tagMarkers> are user-defined.
- It is defined in the *XML Specification* produced by the W3C.
- XML's design goals emphasize transparency, simplicity, and transportability over the Internet.
- Example of XML-based languages include: RSS , Atom, SOAP, and XHTML.
- Several office productivity tools default to XML format for internal data storage. Example: Microsoft Office, OpenOffice.org, and Apple's iWork.

XML Data

How is XML used?

1. XML is used for **defining** and **documenting** object classes.

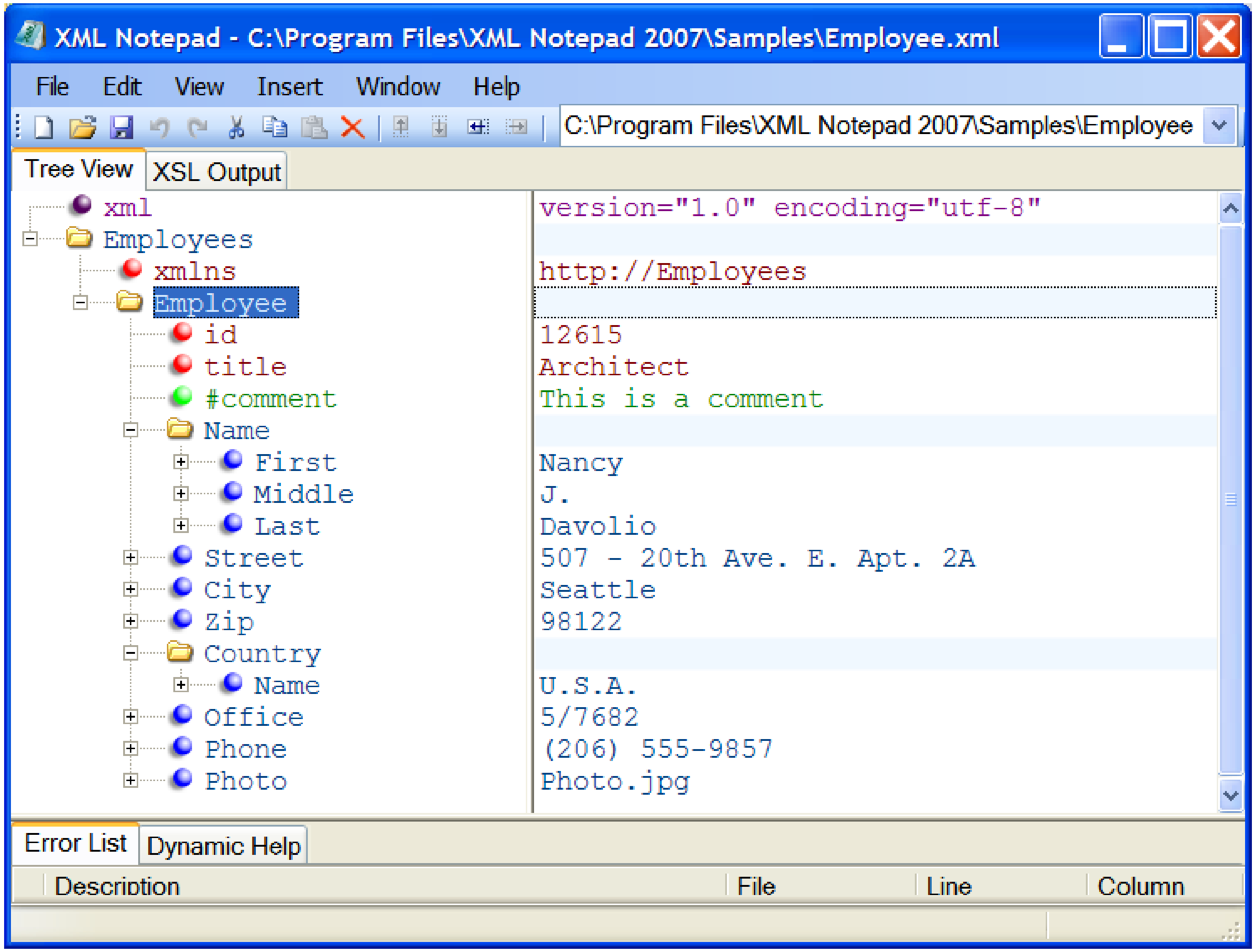
2. For example, an XML document (**.xml**) might contain a collection of complex **employee** elements, such as
   `<employee id="…" title="…" >...</employee>`
   which lexically includes an "**id**" and **"title**" attributes.

3. Employee may also hold other inner elements such as **"name**", "**country", "city",** and **"zip".**

4. An XML-Data schema (**.xsd**) can describe such syntax.

XML Data

How is XML used? – Employee Example


XML Notepad - C:\Program Files\XML Notepad 2007\Samples\Employee.xml
File Edit View Insert Window Help
C:\Program Files\XML Notepad 2007\Samples\Employee
Tree View
XSL Output
Error List
Dynamic Help
Description
File
Line
Column

```
xml                 version="1.0" encoding="utf-8"
Employees
  xmlns             http://Employees
  Employee
    id              12615
    title           Architect
    #comment        This is a comment
    Name
      First         Nancy
      Middle        J.
      Last          Davolio
    Street          507 - 20th Ave. E. Apt. 2A
    City            Seattle
    Zip             98122
    Country
      Name          U.S.A.
    Office          5/7682
    Phone           (206) 555-9857
    Photo           Photo.jpg
```

Microsoft
XML Notepad

XML Data

Sample1. Employee.xml

```
<?xml version="1.0" encoding="utf8" ?>
<Employees xmlns="http://Employees">
    <Employee   id="12615" title="Architect">
        <! This is a comment  >
        <Name>
            <First>Nancy</First>
            <Middle>J.</Middle>
            <Last>Davolio</Last>
        </Name>
        <Street>507  20th Ave. E. Apt. 2A</Street>
        <City>Seattle</City>
        <Zip>98122</Zip>
        <Country>
            <Name>U.S.A.</Name>
        </Country>
        <Office>5/7682</Office>
        <Phone>(206) 5559857</Phone>
        <Photo>Photo.jpg</Photo>
    </Employee>
    <Employee>
        . . .
    </Employee>
</Employees>
```

Attributes: **id, title**

Element: **Street**

Example taken from:
Microsoft XmlNotepad 2007
http://www.microsoft.com/downloads/en/details.aspx?familyid=72d6aa49787d4118ba5f4f30fe913628&displaylang=en

XML Data

Sample1. Employee.xsd – Schema Definition *(fragment)*

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified" targetNamespace="http://Employees" xmlns="http://Employees">

 <xs:complexType name="Country">
 <xs:sequence>
   <xs:element name="Name" type="xs:string" default="U.S.A." />
   </xs:sequence>
 <xs:attribute name="code" type="xs:language">
 <xs:annotation>
   <xs:documentation>The registered IANA country code of the format xxxx. For example: enus.</xs:documentation>
   </xs:annotation>
   </xs:attribute>
   </xs:complexType>

 <xs:simpleType name="City">
 <xs:restriction base="xs:string">
   <xs:minLength value="1" />
   <xs:maxLength value="50" />
   </xs:restriction>
   </xs:simpleType>

 <xs:simpleType name="Zip">
 <xs:restriction base="xs:positiveInteger">
   <xs:maxInclusive value="99999" />
   <xs:minInclusive value="00001" />
   </xs:restriction>
   </xs:simpleType>

 <xs:simpleType name="EmployeeID">
 <xs:annotation>
   <xs:documentation>The ITG assigned 5 digit employee identification</xs:documentation>
   </xs:annotation>
 <xs:restriction base="xs:string">
   <xs:length value="5" />
   </xs:restriction>
   </xs:simpleType>
```

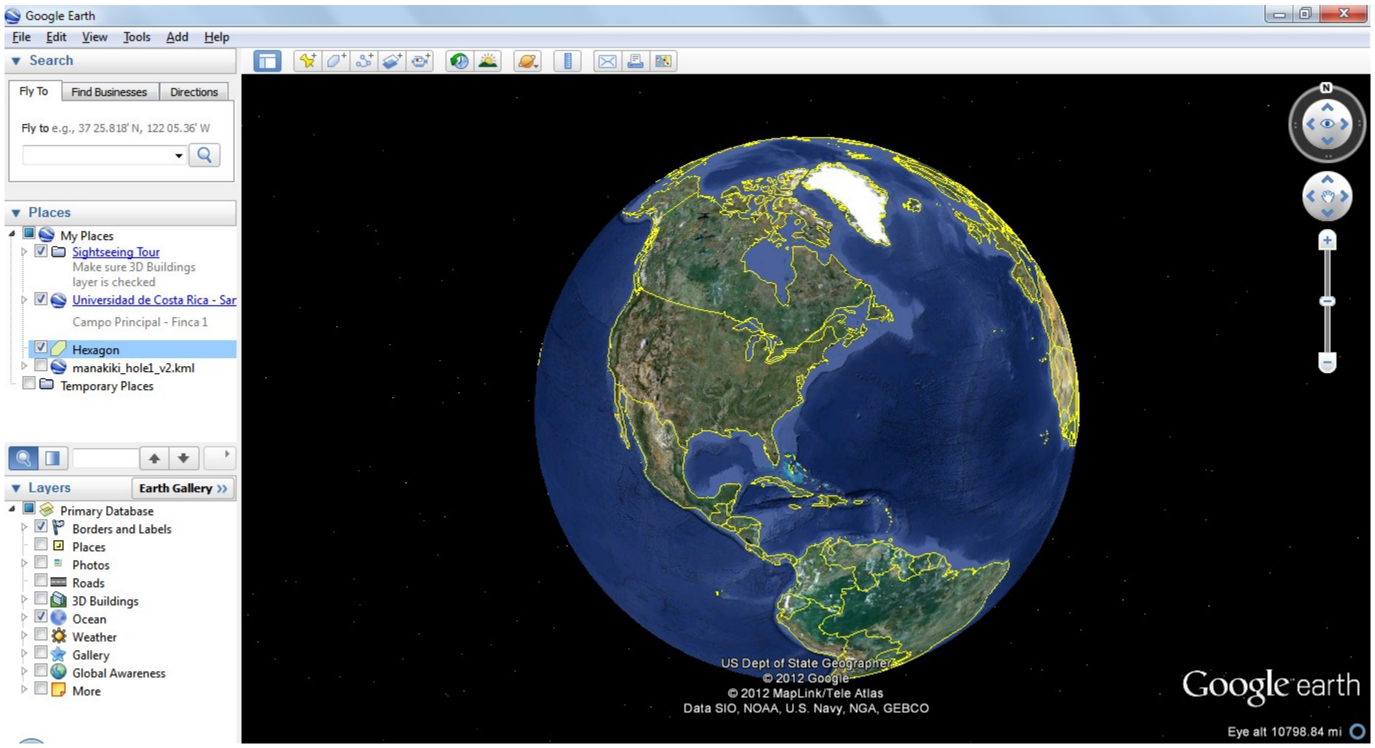
Only a fragment. Lines removed

XML Data

Reference: http://code.google.com/apis/kml/documentation/kml_tut.html

KML is a file format used to display geographic data in an Earth browser such as:

- Google Earth,
- Google Maps, and
- Google Maps for mobile



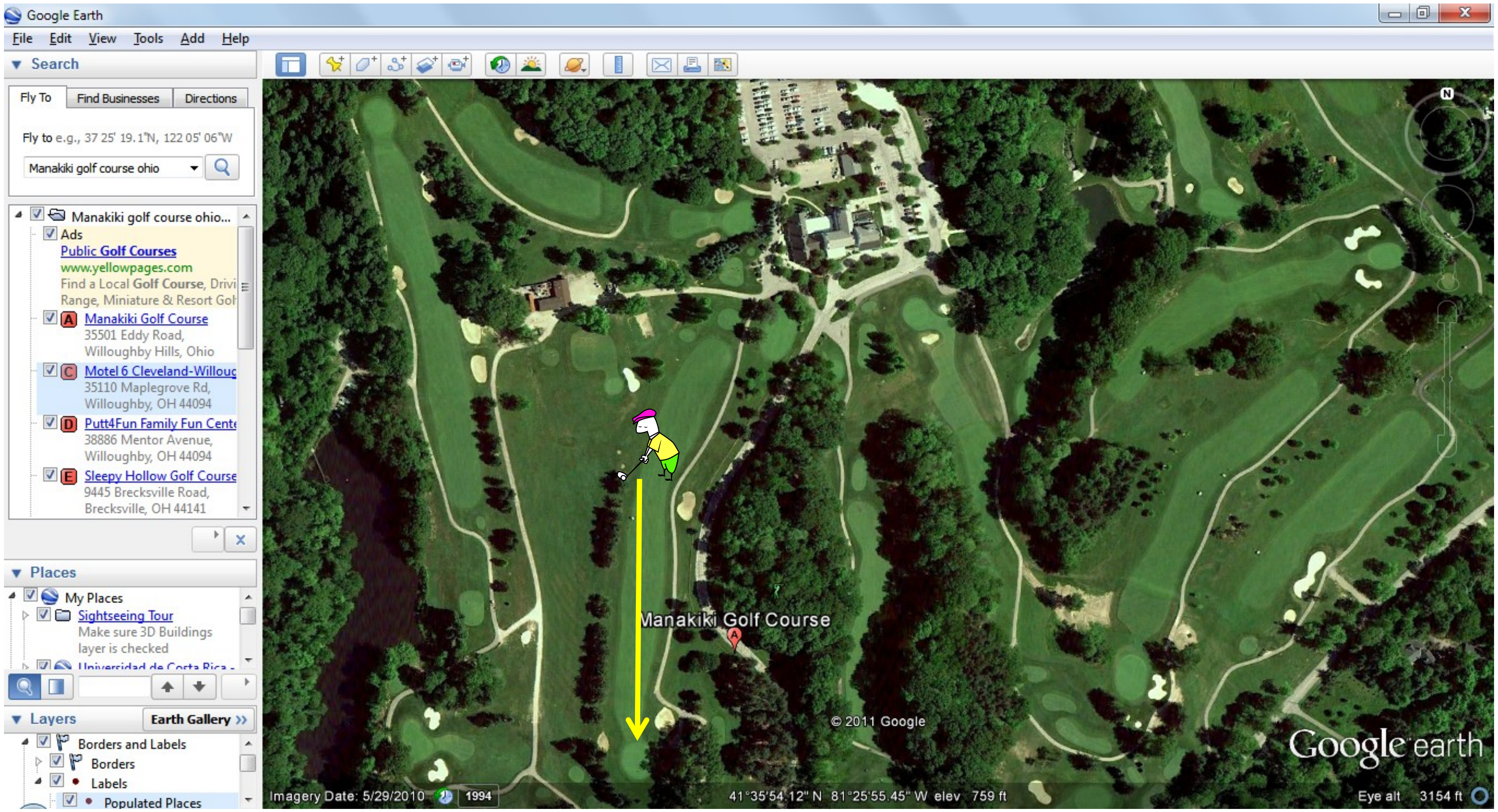
XML Data

Sample 2. Mapping with KML *(fragment)*

```
<?xml version="1.0" encoding=“utf-8" ?>
 <kml xmlns="http://www.opengis.net/kml/2.2">
 <Document>

          <gcPlace gcName="Manakiki Golf Course" gcCity="Willoughby Hills" gcState="Ohio" />

         <Placemark>
                  <name   par="4"   yards="390" >Tee  Hole 1</name>
                 <Point>
                  <coordinates>81.4324182271957,41.5984273639879,0</coordinates>
                  </Point>
          </Placemark>

         <Placemark>
                  <name>Front of Green  Hole 1</name>
                 <Point>
                  <coordinates>81.433182656765,41.5955730479591,0</coordinates>
                  </Point>
           </Placemark>

          <Placemark>
                  <name>Middle of Green  Hole 1</name>
                 <Point>
                  <coordinates>81.4331665635109,41.5954647298964,0</coordinates>
                  </Point>
           </Placemark>

    </Document>
    </kml>
```

XML Data

Sample 2. Mapping with KML *(View from Google Earth)*



Manakiki golf course ohio...
Ads
Public Golf Courses
www.yellowpages.com
Find a Local Golf Course, Drivi
Range, Miniature & Resort Gol

A Manakiki Golf Course
35501 Eddy Road,
Willoughby Hills, Ohio

C Motel 6 Cleveland-Willoug
35110 Maplegrove Rd,
Willoughby, OH 44094

D Putt4Fun Family Fun Cente
38886 Mentor Avenue,
Willoughby, OH 44094

E Sleepy Hollow Golf Course
9445 Brecksville Road,
Brecksville, OH 44141

My Places
Sightseeing Tour
Make sure 3D Buildings
layer is checked

Borders and Labels
Borders
Labels
Populated Places

Imagery Date: 5/29/2010 1994   41°35'54.12" N 81°25'55.45" W elev 759 ft   Eye alt 3154 ft

Reference: http://code.google.com/apis/kml/documentation/kml_tut.html

XML Data

Sample 2. Mapping with KML & Playing Golf

Typical Distances for (Good) Amateur Players

Club	Men	Women
Driver	200-230-260	150-175-200
3-wood	180-215-235	125-150-180
2-Hybrid	170-195-210	105-135-170
3-Hybrid	160-180-200	100-125-160
4-iron	150-170-185	90-120-150
5-iron	140-160-170	80-110-140
6-iron	130-150-160	70-100-130
7-iron	120-140-150	65-90-120
8-iron	110-130-140	60-80-110
9-iron	95-115-130	55-70-95
PW	80-105-120	50-60-80
SW	60-80-100	40-50-60

"Don't worry, Fenton...
I've got the **perfect** club."

Reference: Cartoon by Lash Leroux available at http://www.golfun.net/lash3.htm

XML Data

SAX
Simple API for XML

Strategies for Reading/Parsing an XML File

- Several approaches are available
- Here we will explore two options:

OPTION 1 A SAX (Simple API for XML) XmlPullParser	OPTION 2 W3C Document Builder
You traverse the document programmatically looking for the beginning and ending of element tags, and internal attributes.	A Document Builder object dissects the nodes of the XML document. Resulting lists are managed as Java ArrayLists

References:
http://www.saxproject.org/
http://www.w3.org/DOM/

Note:
The World Wide Web Consortium (W3C.org) is an "international community that develops open standards to ensure the long-term growth of the Web".

XML Data

SAX
Simple API for XML

Example 1. Reading/Parsing a Resource XML File *(code)*

- In this example we will read an XML file saved in the **/re/xml** folder. The example file contains a set of place-markers around a golf course.

- A **SAX** (Simple API for XML) **XmlPullParser** traverses the document using the **.next()** method and detects the following main *eventTypes*

  START_TAG
  TEXT
  END_TAG
  END_DOCUMENT

- When the beginning of a tag is recognized, we use the **.getName()** method to grab the tag name.

- We use the method **.getText()** to extract data after TEXT event.

Reference: http://www.saxproject.org/

XML Data

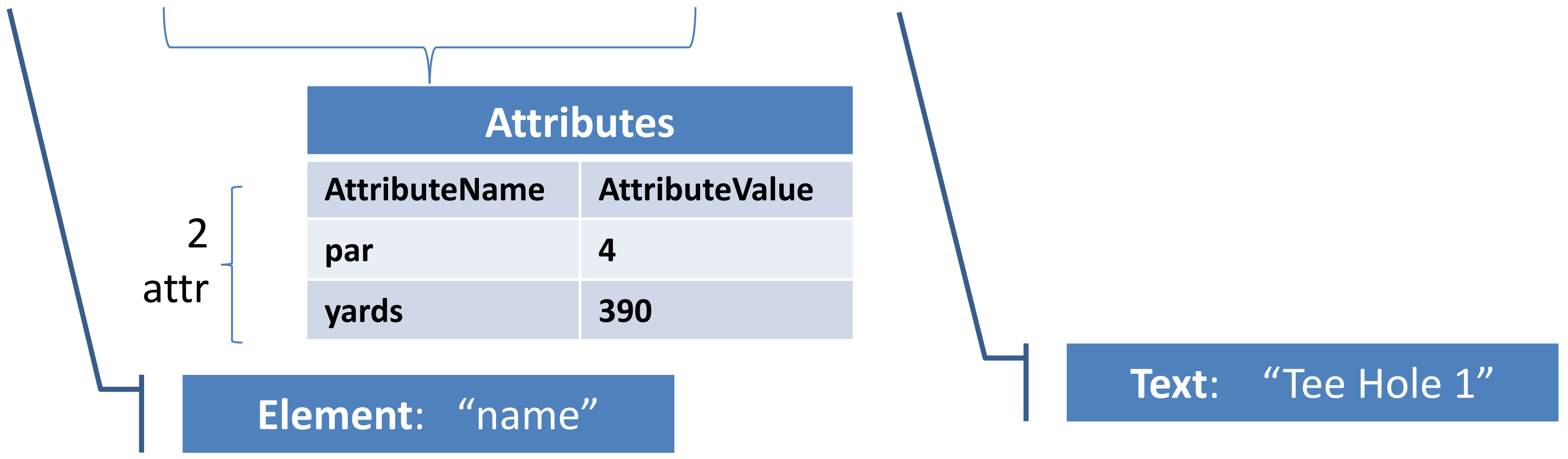
Example1. SAX - Reading/Parsing a Resource KML File

Attributes from an <**element>** can be extracted using the methods:

```
.getAttributeCount()
.getAttributeName()
.getAttributeValue()
```

Consider the **name**-element in the example below:

```
<name par="4" yards="390" >Tee Hole 1</name>
```



Attributes

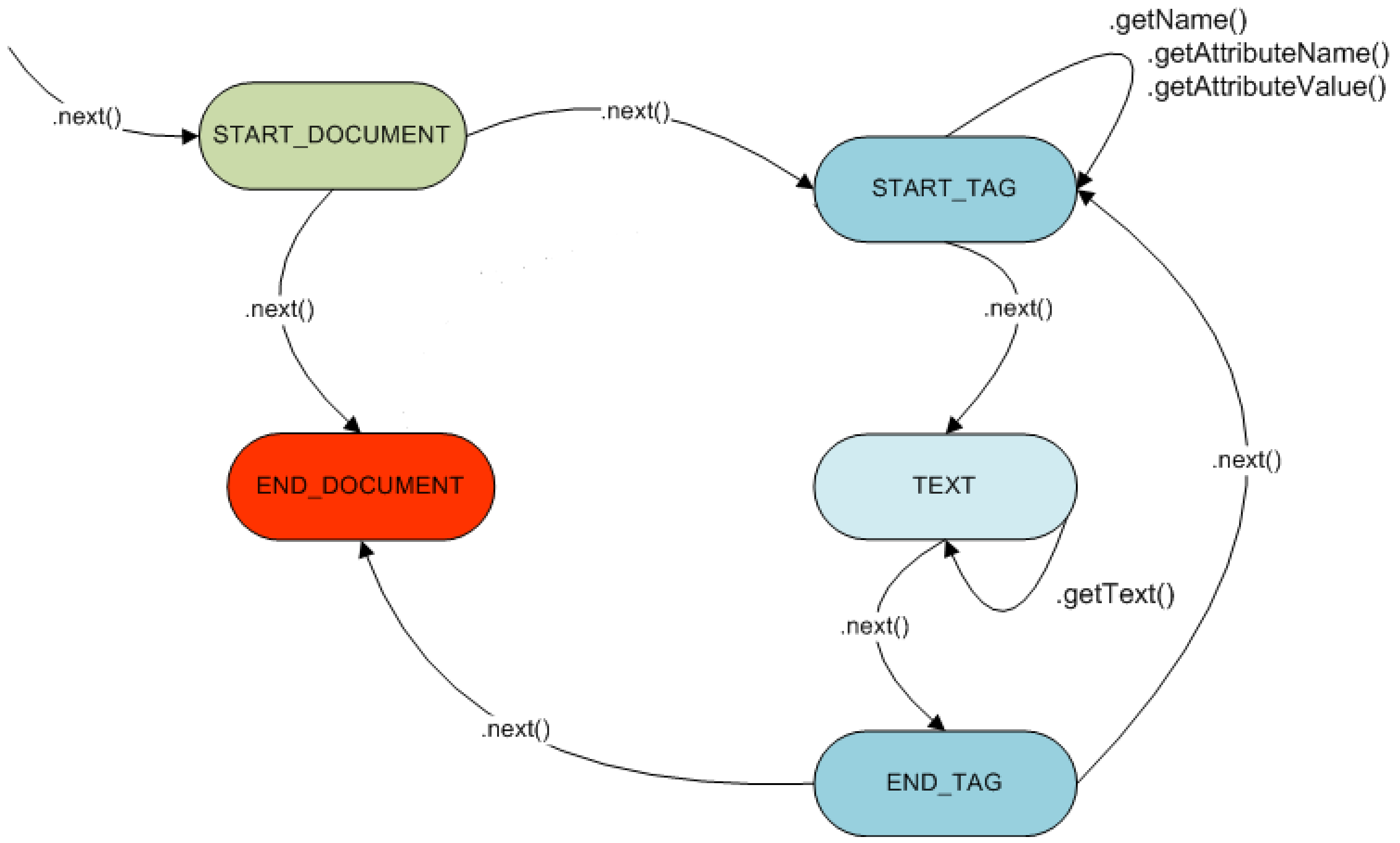
AttributeName	AttributeValue
par	4
yards	390

XML Data

SAX
Simple API for XML

Example 1. SAX - Reading/Parsing a Resource KML File

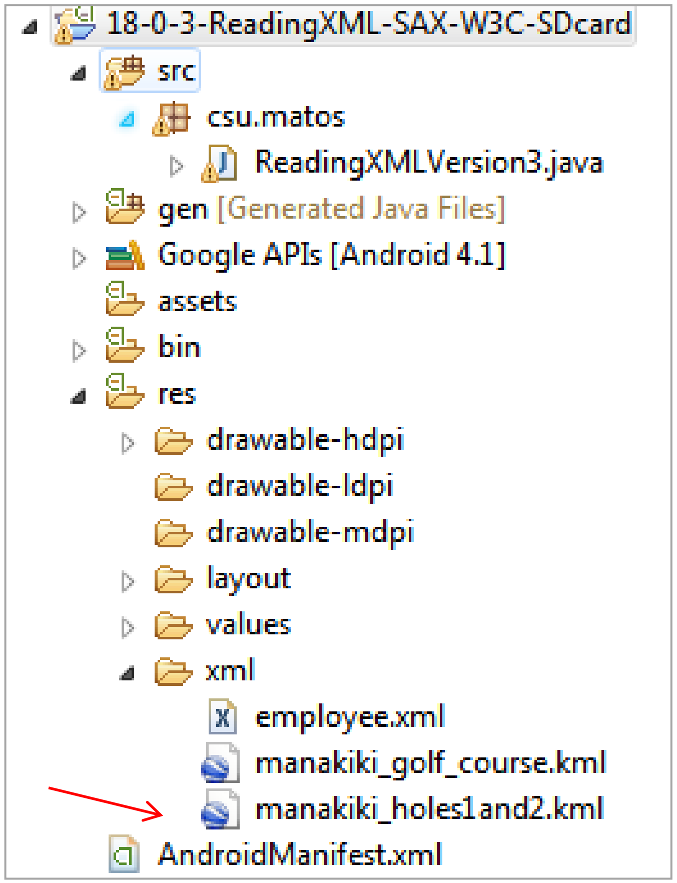
Using the **XmlPullParser** class to generate scanner/parser to traverse an XML document



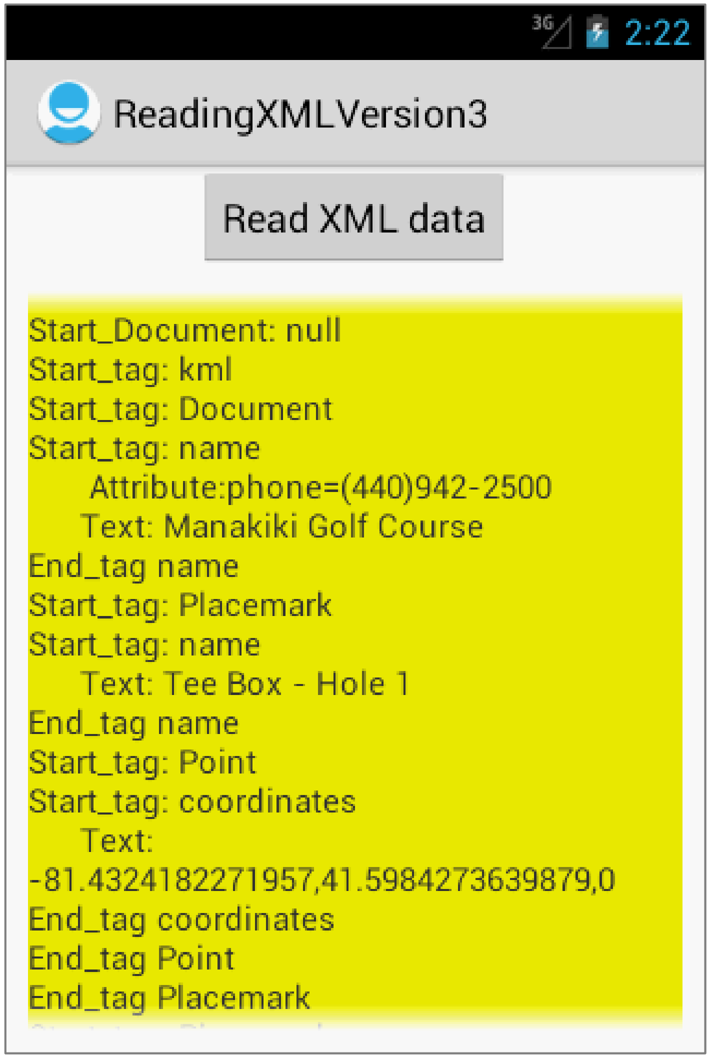
XML Data

SAX
Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*



```
18-0-3-ReadingXML-SAX-W3C-SDcard
  src
    csu.matos
      ReadingXMLVersion3.java
  gen [Generated Java Files]
  Google APIs [Android 4.1]
  assets
  bin
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    layout
    values
    xml
      employee.xml
      manakiki_golf_course.kml
      manakiki_holes1and2.kml
  AndroidManifest.xml
```



```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
      Attribute:phone=(440)942-2500
     Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
     Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
     Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

XML Data

SAX
Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name  phone="(440)942-2500" >Manakiki Golf Course</name>
    <Placemark>
      <name>Tee Box - Hole 1</name>
      <Point>
        <coordinates>-81.4324182271957,41.5984273639879,0</coordinates>
      </Point>
    </Placemark>
    <Placemark>
      <name>Front of Green - Hole 1</name>
      <Point>
        <coordinates>-81.433182656765,41.5955730479591,0</coordinates>
      </Point>
    </Placemark>
    <Placemark>
      <name>Middle of Green - Hole 1</name>
      <Point>
        <coordinates>-81.4331665635109,41.5954647298964,0</coordinates>
      </Point>
    </Placemark>
           . . . (edited for brevity) ...
  </Document>
</kml>
```

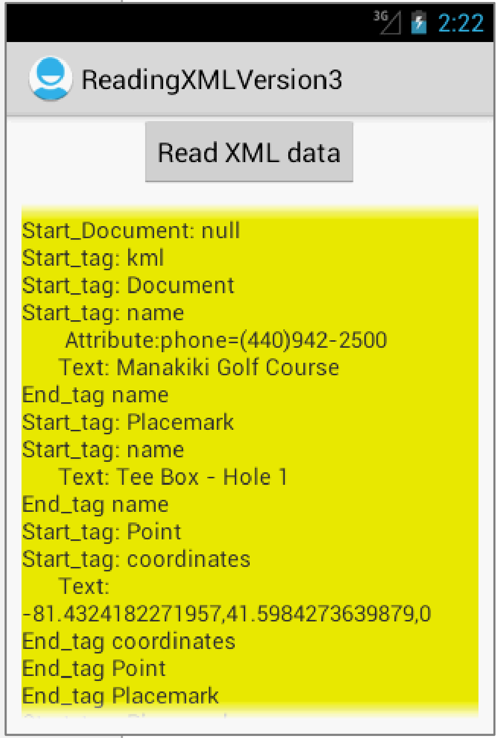
This is an abbreviated version of Sample 2

XML Data

SAX
Simple API for XML

Example 1 . SAX - Reading a Resource KML File *(code)*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btnReadXml"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Read XML data" />

    <ScrollView
        android:id="@+id/ScrollView01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:padding="10dp">

        <TextView
            android:id="@+id/txtMsg"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="#ffeeee00"  />
    </ScrollView>

</LinearLayout>
```

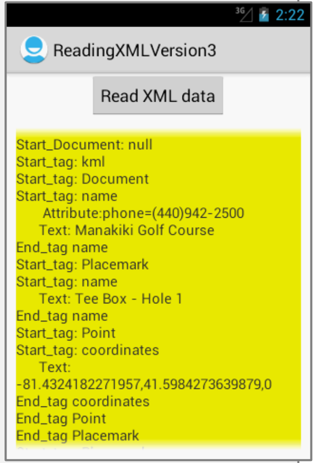


```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
     Attribute:phone=(440)942-2500
     Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
     Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
     Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

SAX
Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```
public class ReadingXMLVersion3 extends Activity {

   Button btnGo;
   TextView txtMsg;

   XmlPullParser parser;
   String elementName;
   String attributeName;
   String attributeValue;
   StringBuilder str;



   String holeNo = "";
   String attrParName = "";
   String attrParValue = "";
   String attrHandicapName = "";
   String attrHandicapValue = "";
   String golfCourseName = "";
```



```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
    Attribute:phone=(440)942-2500
   Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
   Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
   Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

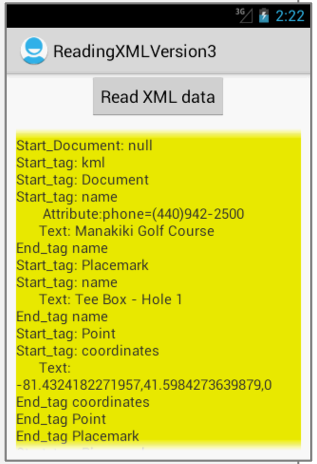
XML Data

SAX

Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   txtMsg = (TextView) findViewById(R.id.txtMsg);

   btnGo = (Button) findViewById(R.id.btnReadXml);
   btnGo.setFocusable(true);
   btnGo.requestFocus();
   btnGo.setOnClickListener(new OnClickListener() {

      @Override
      public void onClick(View v) {
         txtMsg.setTextSize(15);

         useSaxOrgXmlPullParser(); // see www.saxproject.org
         //useW3cOrgDocumentBuilder(); // see www.w3c.org
      }
   });

}// onCreate
```



```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
    Attribute:phone=(440)942-2500
    Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
    Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
    Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

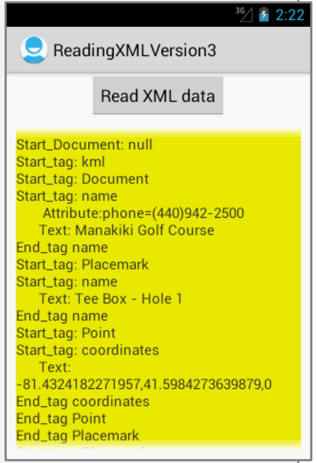
XML Data

SAX

Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```
public void useSaxOrgXmlPullParser() {

   parser = getResources().getXml(R.xml.manakiki_holes1and2);
   str = new StringBuilder();
   txtMsg.setText("");

   int eventType = -1;
   try {
      while (eventType != XmlPullParser.END_DOCUMENT) {

         eventType = parser.next();

         if (eventType == XmlPullParser.START_DOCUMENT) {
            elementName = parser.getName();
            str.append("\nStart_Document: " + elementName);
            getInnerAttributes();

         } else if (eventType == XmlPullParser.END_DOCUMENT) {
            str.append("\nEnd_document");

         } else if (eventType == XmlPullParser.START_TAG) {
            elementName = parser.getName();
            str.append("\nStart_tag: " + elementName);
            getInnerAttributes();
```

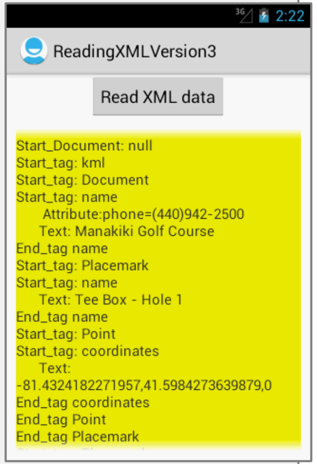


```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
      Attribute:phone=(440)942-2500
     Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
     Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
     Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

SAX
Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```
        } else if (eventType == XmlPullParser.END_TAG) {
           elementName = parser.getName();
           str.append("\nEnd_tag " + elementName);

        } else if (eventType == XmlPullParser.TEXT) {
           String text = parser.getText();
           str.append("\n\t Text:  " + text);

        } else {
           str.append("\n<<< UNEXEPCTED EVENT >>>");
        }
     }
     txtMsg.setText(str.toString());

   } catch (XmlPullParserException e) {

     e.printStackTrace();
   } catch (IOException e) {

     e.printStackTrace();
   }
}// useXmlPullParser
```

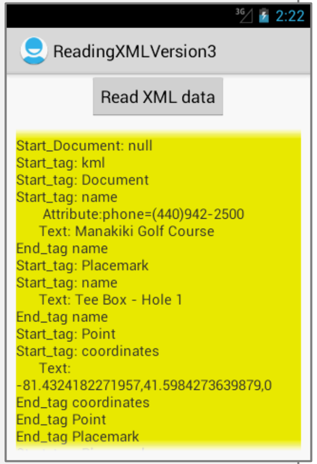


```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
      Attribute:phone=(440)942-2500
     Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
     Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
     Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

SAX
Simple API for XML

Example 1. SAX - Reading a Resource KML File *(code)*

```java
public void getInnerAttributes() {

   for (int i = 0; i < parser.getAttributeCount(); i++) {

      attributeName = parser.getAttributeName(i);

      attributeValue = parser.getAttributeValue(i);

      str.append("\n\t  Attribute:"
                + attributeName + "=" + attributeValue);
   }

}
```



```
Start_Document: null
Start_tag: kml
Start_tag: Document
Start_tag: name
     Attribute:phone=(440)942-2500
    Text: Manakiki Golf Course
End_tag name
Start_tag: Placemark
Start_tag: name
    Text: Tee Box - Hole 1
End_tag name
Start_tag: Point
Start_tag: coordinates
    Text:
-81.4324182271957,41.5984273639879,0
End_tag coordinates
End_tag Point
End_tag Placemark
```

XML Data

Example 2. W3C DOM - Reading XML (from the SD card)

In this example we read the same XML data of the previous example, but rather than stepping through the SAX *eventTypes* recognized by the XmlPullParser,

1. A W3C **DocumentBuilder** parser will be used.
2. The new parser requires less code than a SAX recognizer.
3. The input **XML** file is converted by the W3C parser into an equivalent **tree**.
4. The input file is stored externally in the SD card

XML Data

Example 2. W3C - Reading XML from the SD card

Elements from the input XML file are represented in the output parse-tree as **NodeLists.**

<Document> acts as the root of the tree.

PHASE 1. For each XML element construct a NodeList (ArrayList-like collection) using the **.getElementsByTagName()** method.

PHASE2. Explore an individual node from a NodeList using the methods:

```
.item(i),
.getName() ,
.getValue() ,
.getFirstChild() ,
.getAttributes(), etc.
```

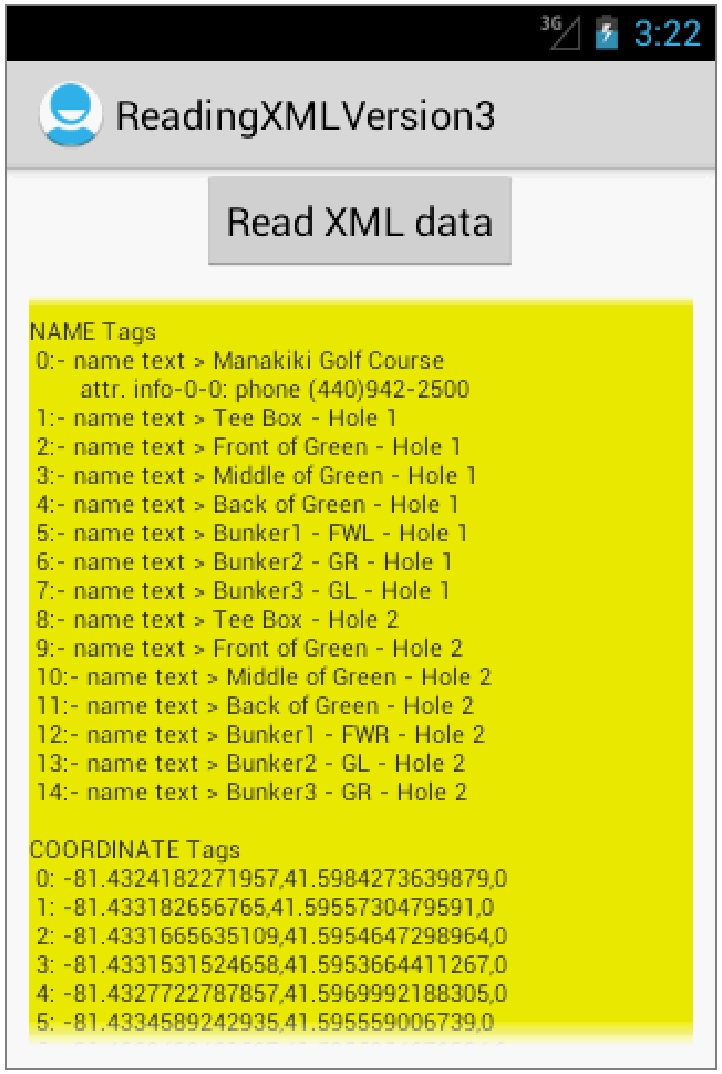
XML Data

Example 2. W3C - Reading an XML file from the SD card

This code is continuation of the previous example (you need to uncomment call in the onCreate method)

'name' NodeList

'coordinate ' NodeList



```
NAME Tags
 0:- name text > Manakiki Golf Course
       attr. info-0-0: phone (440)942-2500
 1:- name text > Tee Box - Hole 1
 2:- name text > Front of Green - Hole 1
 3:- name text > Middle of Green - Hole 1
 4:- name text > Back of Green - Hole 1
 5:- name text > Bunker1 - FWL - Hole 1
 6:- name text > Bunker2 - GR - Hole 1
 7:- name text > Bunker3 - GL - Hole 1
 8:- name text > Tee Box - Hole 2
 9:- name text > Front of Green - Hole 2
 10:- name text > Middle of Green - Hole 2
 11:- name text > Back of Green - Hole 2
 12:- name text > Bunker1 - FWR - Hole 2
 13:- name text > Bunker2 - GL - Hole 2
 14:- name text > Bunker3 - GR - Hole 2

COORDINATE Tags
 0: -81.4324182271957,41.5984273639879,0
 1: -81.433182656765,41.5955730479591,0
 2: -81.4331665635109,41.5954647298964,0
 3: -81.4331531524658,41.5953664411267,0
 4: -81.4327722787857,41.5969992188305,0
 5: -81.4334589242935,41.595559006739,0
```

XML Data

Example 2. W3C - Reading an XML file from the SD card

```java
private void useW3cOrgDocumentBuilder() {
try {
     String kmlFile = Environment.getExternalStorageDirectory()
                               .getPath() + "/manakiki_holes1and2.kml";
     InputStream is = new FileInputStream(kmlFile);

     DocumentBuilder docBuilder = DocumentBuilderFactory
                                      .newInstance()
                                      .newDocumentBuilder();

     Document document = docBuilder.parse(is);

     NodeList listNameTag = null;
     NodeList listCoordinateTag = null;
     NodeList listGcDataTag = null;

     if (document == null) {
          txtMsg.setText("Big problems with the parser");
          Log.v("REALLY BAD!!!!", "document was NOT made by parser");
          return;
     }
```

XML Data

Example 2. W3C - Reading an XML file from the SD card

```
     StringBuilder str = new StringBuilder();

     // dealing with 'name' tagged elements
     listNameTag = document.getElementsByTagName("name");

     // showing 'name' tags
     str.append("\nNAME Tags");
     for (int i = 0; i < listNameTag.getLength(); i++) {

          Node node = listNameTag.item(i);
          int size = node.getAttributes().getLength();
          String text = node.getTextContent();
          str.append("\n " + i + ":- name text > " + text);

          // get all attributes of the current element (i-th hole)
          for (int j = 0; j < size; j++) {
               String attrName = node.getAttributes().item(j).getNodeName();
               String attrValue = node.getAttributes().item(j).getNodeValue();

               str.append( "\n\t attr. info-" + i + "-" + j + ": "
                          + attrName + " " + attrValue);
          }
     }
```

XML Data

Example 2. W3C - Reading an XML file from the SD card

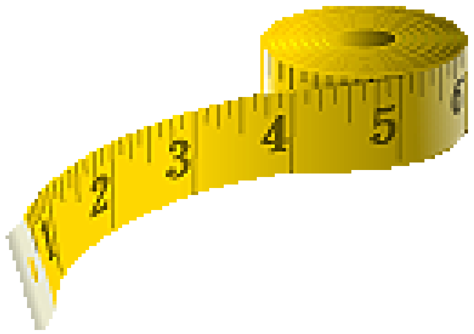
```
     // dealing with 'coordinates' tagged elements
     listCoordinateTag = document.getElementsByTagName("coordinates");

     // showing 'coordinates' tags
     str.append("\n\nCOORDINATE Tags");
     for (int i = 0; i < listCoordinateTag.getLength(); i++) {
          String coordText = listCoordinateTag.item(i).getFirstChild()
                                                       .getNodeValue();
          str.append("\n " + i + ": " + coordText);
     }

     txtMsg.setText(str.toString());

     } catch (FileNotFoundException e) {
         e.printStackTrace();
     } catch (ParserConfigurationException e) {
     e.printStackTrace();
     } catch (SAXException e) {
     e.printStackTrace();
     } catch (IOException e) {
     e.printStackTrace();
     }
}// useW3cOrgDocumentBuilder
```

Reading XML Files

< Questions />

XML Data

Appendix A. Calculating Distance Between Two C



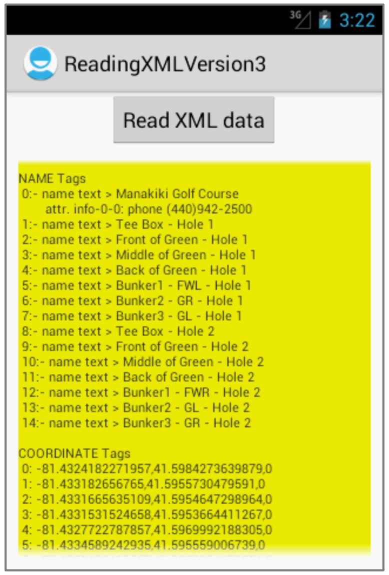
```
    import android.location.Location;

    . . .

    private int distanceYards(GolfMarker gm){
        // calculating distance (yards) between two coordi
        int intDistance = 0;
        double distance = 0;

        Location locationA = new Location("point: Here");
        locationA.setLatitude(Double.parseDouble(aLatitude));
        locationA.setLongitude(Double.parseDouble(aLongitude));

        Location locationB = new Location("point: F/M/B Green");
        locationB.setLatitude(Double.parseDouble(bLatitude));
        locationB.setLongitude(Double.parseDouble(bLongitude));

        distance = locationA.distanceTo(locationB) * METER_TO_YARDS;
        intDistance = (int) Math.round(distance);
        return intDistance;
    }

}// GolfMarker
```

XML Data

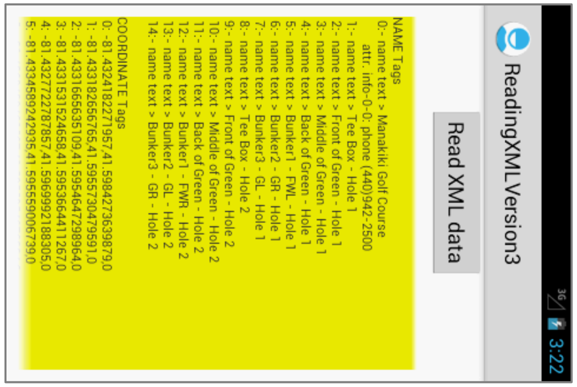
Appendix B. Screen Oriented in Portrait Mode Only

NOTE:
You may want to modify the Manifest to stop (landscape) re-orientation.
Add the following attributes to the <activity … > entry

```
android:screenOrientation="portrait"
android:configChanges="keyboardHidden|orientation"
```



```
NAME Tags
 0:- name text > Manakiki Golf Course
       attr. info-0-0: phone (440)942-2500
 1:- name text > Tee Box - Hole 1
 2:- name text > Front of Green - Hole 1
 3:- name text > Middle of Green - Hole 1
 4:- name text > Back of Green - Hole 1
 5:- name text > Bunker1 - FWL - Hole 1
 6:- name text > Bunker2 - GR - Hole 1
 7:- name text > Bunker3 - GL - Hole 1
 8:- name text > Tee Box - Hole 2
 9:- name text > Front of Green - Hole 2
 10:- name text > Middle of Green - Hole 2
 11:- name text > Back of Green - Hole 2
 12:- name text > Bunker1 - FWR - Hole 2
 13:- name text > Bunker2 - GL - Hole 2
 14:- name text > Bunker3 - GR - Hole 2

COORDINATE Tags
 0: -81.4324182271957,41.5984273639879,0
 1: -81.433182656765,41.5955730479591,0
 2: -81.4331665635109,41.5954647298964,0
 3: -81.4331531524658,41.5953664411267,0
 4: -81.4327722787857,41.5969992188305,0
 5: -81.4334589242935,41.595559006739,0
```



```
NAME Tags
 0:- name text > Manakiki Golf Course
       attr. info-0-0: phone (440)942-2500
 1:- name text > Tee Box - Hole 1
 2:- name text > Front of Green - Hole 1
 3:- name text > Middle of Green - Hole 1
 4:- name text > Back of Green - Hole 1
 5:- name text > Bunker1 - FWL - Hole 1
 6:- name text > Bunker2 - GR - Hole 1
 7:- name text > Bunker3 - GL - Hole 1
 8:- name text > Tee Box - Hole 2
 9:- name text > Front of Green - Hole 2
 10:- name text > Middle of Green - Hole 2
 11:- name text > Back of Green - Hole 2
 12:- name text > Bunker1 - FWR - Hole 2
 13:- name text > Bunker2 - GL - Hole 2
 14:- name text > Bunker3 - GR - Hole 2

COORDINATE Tags
 0: -81.4324182271957,41.5984273639879,0
 1: -81.433182656765,41.5955730479591,0
 2: -81.4331665635109,41.5954647298964,0
 3: -81.4331531524658,41.5953664411267,0
 4: -81.4327722787857,41.5969992188305,0
 5: -81.4334589242935,41.595559006739,0
```