

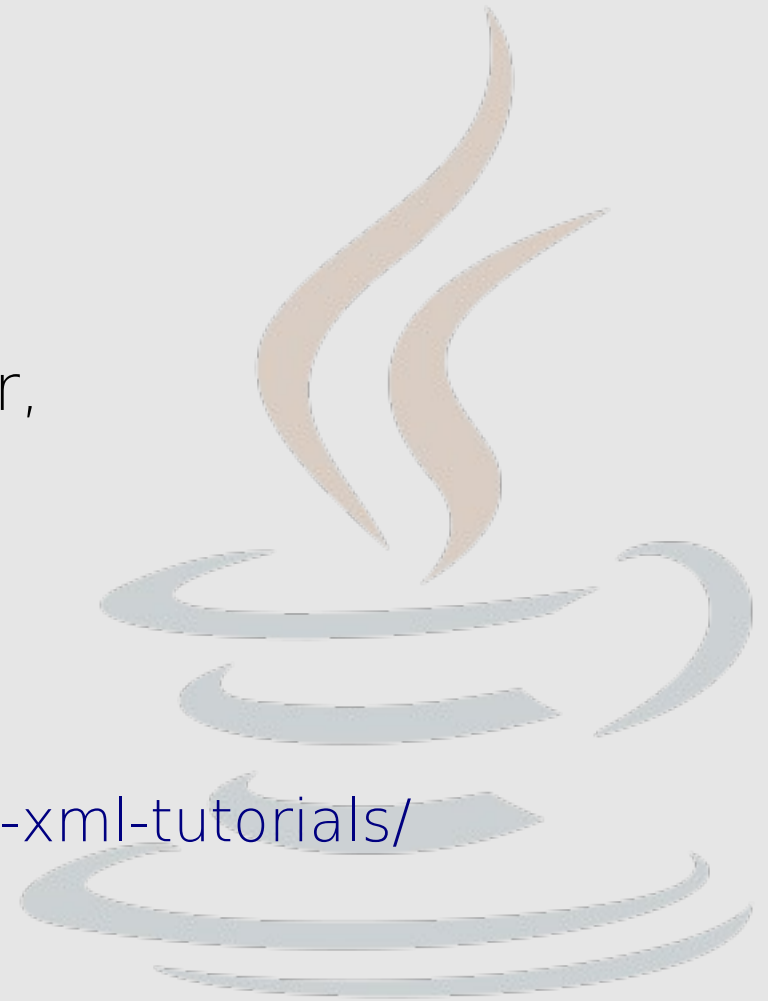
# JAVA | XML

## ZAGADNIENIA:

- DOM,
- SAX,
- JAXB, XMLDecoder i XMLEncoder,
- ANT.

## MATERIAŁY:

<http://www.mkyong.com/tutorials/java-xml-tutorials/>  
<http://ant.apache.org>



# XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">
  <channel>
    <title>Aktualności</title>
    <link>http://www.uj.edu.pl/universytet/aktualnosci/...</link>
    <description>Aktualności Uniwersytetu Jagiellońskiego</description>
    <item>
      <title>Lista Pamięci - Stanisław Szczur</title>
      <link>http://www.uj.edu.pl/universytet/aktualnosci/...</link>
      <description />
      <pubDate>Thu, 20 Dec 2012 07:44:00 GMT</pubDate>
      <dc:creator>Jolanta Herian-Ślusarska</dc:creator>
      <dc:date>2012-12-20T07:44:00Z</dc:date>
    </item>
    <item>...</item>
    ...
  </channel>
</rss>
```

# DOM – ODCZYT

Parsery DOM (*Document Object Model*) tworzą drzewo reprezentujące dane zawarte w dokumencie XML. Po zbudowaniu DOM przetwarzanie odbywa się na modelu w pamięci operacyjnej.


```
import java.io.IOException;
import java.net.URL;

import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

public class DOMExample {

    public static void main(String[] args) throws
        ParserConfigurationException, SAXException, IOException {
```



# DOM – ODCZYT

```
URL url = new URL(
"http://www.uj.edu.pl/universytet/aktualnosci/-/journal/rss/10172/36018?
doAsGroupId=10172&refererPlid=10181"); // jakis RSS

// domyslna fabryka "obiektow tworzących dokumenty"
DocumentBuilderFactory dbFactory =
    DocumentBuilderFactory.newInstance();

// obiekt "tworzący dokumenty"
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

// tworzenie modelu DOM na podstawie źródła XML (parsowanie XML'a)
Document doc = dBuilder.parse(url.openStream());

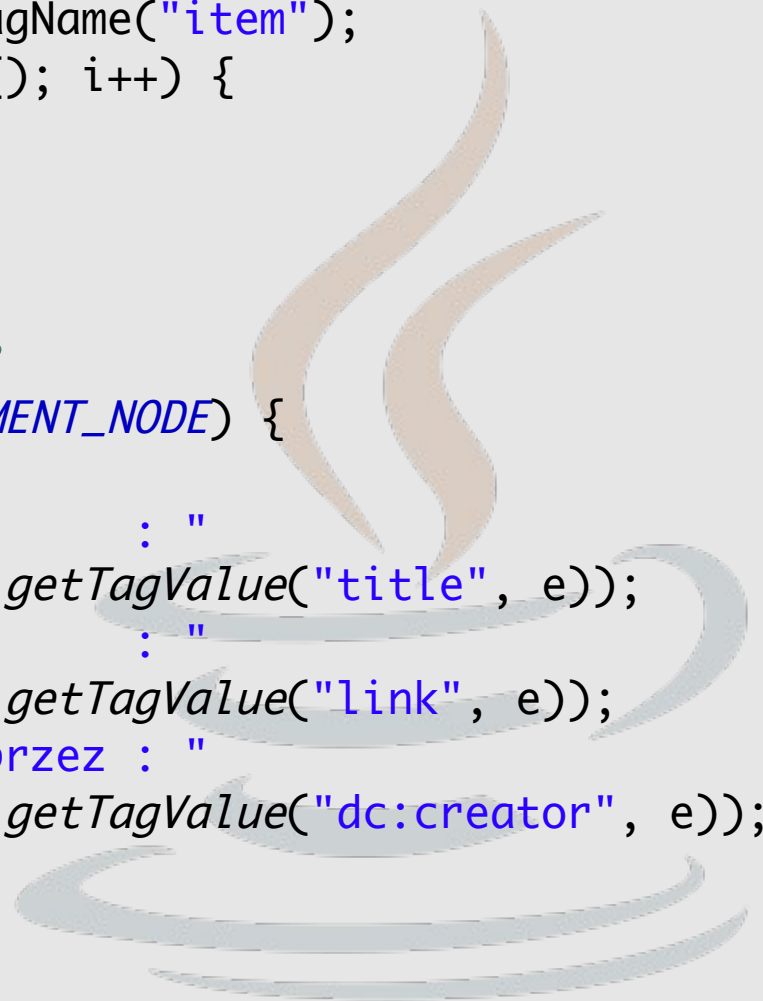
// obiekt doc umożliwia dostęp do wszystkich danych zawartych
// w dokumencie XML
System.out.println("Root element :" +
    doc.getDocumentElement().getNodeName());
```

# DOM – ODCZYT

```
NodeList nList = doc.getElementsByTagName("item");
for (int i = 0; i < nList.getLength(); i++) {

    // lista "dzieci" i-tego itema
    Node n = nList.item(i);

    // czy "dziecko" jest elementem?
    if (n.getNodeType() == Node.ELEMENT_NODE) {
        Element e = (Element) n;
        System.out.println("Tytuł          : "
                           + getTagValue("title", e));
        System.out.println("Link          : "
                           + getTagValue("link", e));
        System.out.println("dodane przez : "
                           + getTagValue("dc:creator", e));
    }
}
```



# DOM – ODCZYT

// zwraca wartosc zapisana w tagu s wewnatrz elementu e  
**private static** String getTagValue(String s, Element e) {

// lista “dzieci” e o nazwie s  
 NodeList nl = e.getElementsByTagName(s)  
 // pierwszy wpis z tej listy

// to co on zawiera – jego “dzieci”

// pierwsze z tych dzieci  
 Node n = (Node) nl.item(0);  
 // zawartosc, ktora tam jest  
 **return** n.getNodeValue();

}

}

.item(0)

.getChildNodes();

# DOM – MODYFIKACJA

Możliwa jest także modyfikacja zawartości DOM utworzonego w wyniku parsowania dokumentu XML. Przykładowo:

```
// pobieramy element title i zmieniamy go
if ("title".equals(node.getNodeName())) {
    node.setTextContent("Nowy tytuł");
}
// kasujemy link
if ("link".equals(node.getNodeName())) {
    itemElement.removeChild(node);
}

// tworzenie nowego dokumentu
Document doc = docBuilder.newDocument();
Element e = doc.createElement("root");
doc.appendChild(e);
```



# DOM – ZAPIS

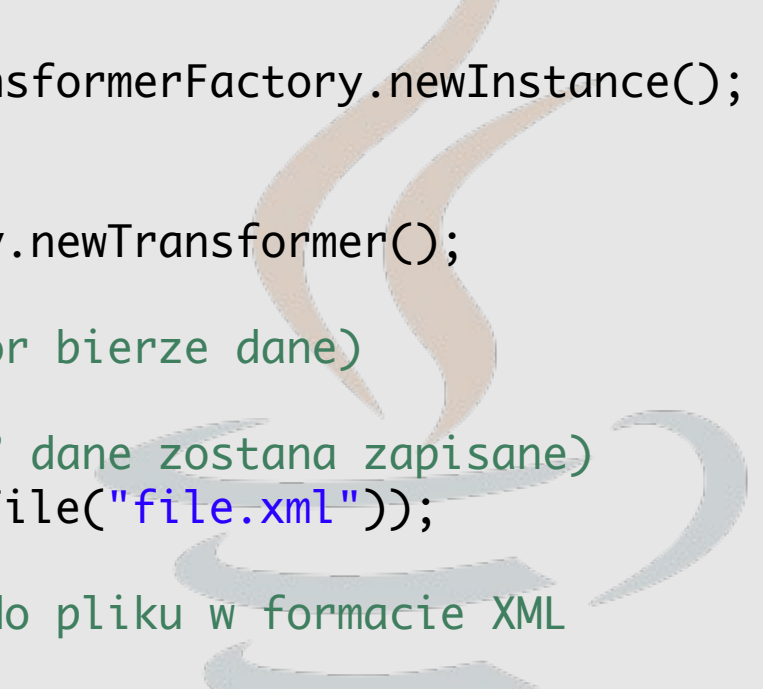
Zapis dokumentu:

```
// domyslna fabryka transformatorow
TransformerFactory transformerFactory = TransformerFactory.newInstance();

// nowy transformator
Transformer transformer = transformerFactory.newTransformer();

// wejscie transformatora (skad transformator bierze dane)
DOMSource source = new DOMSource(doc);
// wyjscie transformatora (gdzie "zmienione" dane zostana zapisane)
StreamResult result = new StreamResult(new File("file.xml"));

// uruchomienie transformatora – zapis DOM do pliku w formacie XML
transformer.transform(source, result);
```





# SAX – ODCZYT

SAX (*Simple API for XML*) w miarę czytania dokumentu wywołuje zdarzenia związane z parsowaniem. Parsery SAX są szybsze i nie wymagają tak dużej ilości pamięci jak DOM.

```
import java.io.IOException;
import java.net.URL;
import javax.xml.parsers.*;
import org.xml.sax.*;
public class SAXExample {
    public static void main(String[] args) throws SAXException,
        IOException, ParserConfigurationException{
        URL url = new URL("http://www.uj.edu.pl/...");
        SAXParserFactory f = SAXParserFactory.newInstance();
        SAXParser saxParser = f.newSAXParser();
        DefaultHandler handler = new ExampleSAXHandler();
        saxParser.parse(url.openStream(), handler);
    }
}
```

# DefaultHandler

- **void characters(char[] ch, int start, int length)** – wywoływane przy odczycie znaku wewnątrz elementu,
- **void endDocument()** – wywoływane gdy koniec dokumentu
- **void endElement(String uri, String localName, String qName)** – koniec elementu,
- **void error(SAXParseException e)** – błąd (możliwy do naprawienia),
- **void fatalError(SAXParseException e)** – błąd,
- **void ignorableWhitespace(char[] ch, int start, int length)** – ignorowany pusty znak,
- **void startDocument()** – początek dokumentu,
- **void startElement(String uri, String localName, String qName, Attributes attributes)** – początek elementu
- **void warning(SAXParseException e)** – ostrzeżenie parsera.

...

# DefaultHandler

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

class ExampleSAXHandler extends DefaultHandler {

    public void startElement(String uri, String localName, String qName,
                             Attributes attributes) throws SAXException {
        System.out.println("Element :" + qName);
    }
    public void endElement(String uri, String localName, String qName)
                             throws SAXException {
        System.out.println("Koniec elementu :" + qName);
    }
    public void characters(char ch[], int start, int length)
                             throws SAXException {
        System.out.println("zawartosc : " + new String(ch, start, length));
    }
}
```

# JAXB

JAXB (*Java Architecture for XML Binding*) to standard serializacji XML dla obiektów Javy. Został on zintegrowany z JDK/JRE od wersji 1.6

```
import java.io.File;

import javax.xml.bind.*;
import javax.xml.bind.annotation.*;

@XmlRootElement
class Person {

    String name;
    int age;

    public String getName() {
        return name;
    }
}
```



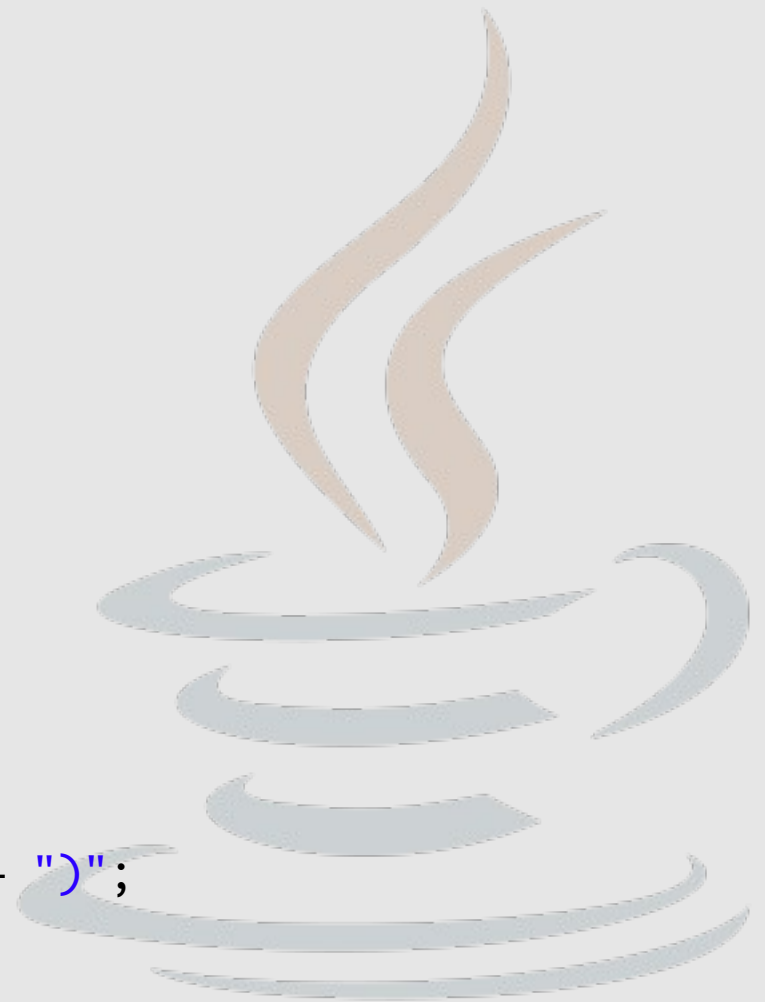
# JAXB

```
@XmlElement
public void setName(String name) {
    this.name = name;
}

public int getAge() {
    return age;
}

@XmlElement
public void setAge(int age) {
    this.age = age;
}

public void toString() {
    return this.name + " (" + this.age + ")";
}
}
```

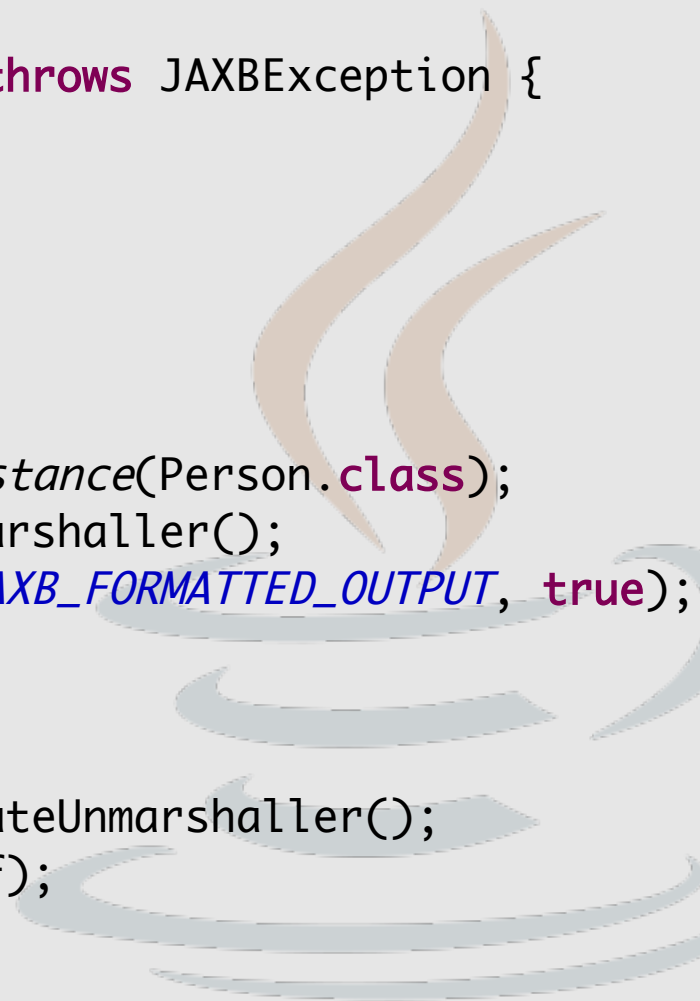


# JAXB

```
public class JAXBExample{
    public static void main(String[] args) throws JAXBException {

        Person p = new Person();
        p.setName("Barnaba");
        p.setAge(33);

        File f = new File("person.xml");
        JAXBContext ctx = JAXBContext.newInstance(Person.class);
        Marshaller marshaller = ctx.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        marshaller.marshal(p, System.out);
        marshaller.marshal(p, f);
        p = null;
        Unmarshaller unmarshaller = ctx.createUnmarshaller();
        p = (Person)unmarshaller.unmarshal(f);
        System.out.println(p);
    }
}
```



# SERIALIZACJA I XML

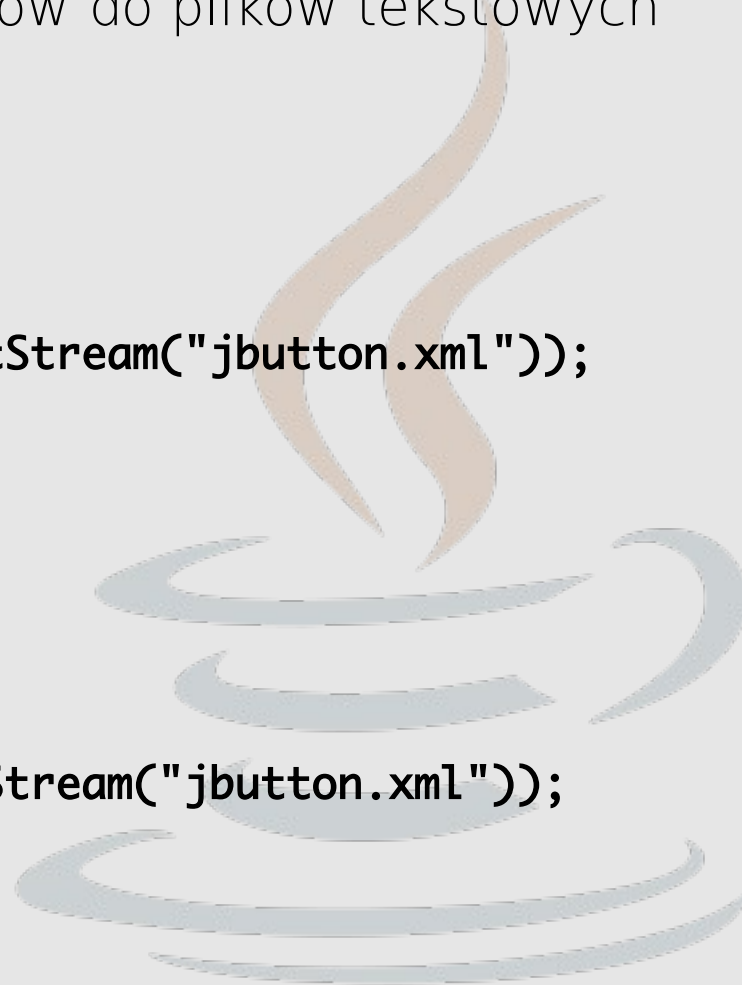
Inna metoda serializacji niektórych obiektów do plików tekstowych w formacie XML:

## XMLEncoder:

```
XMLEncoder e = new XMLEncoder(new FileOutputStream("jbutton.xml"));  
e.writeObject(new JButton("Hello world"));  
e.close();
```

## XMLDecoder:

```
XMLDecoder d = new XMLDecoder(new FileInputStream("jbutton.xml"));  
obj = d.readObject();  
d.close();
```



# ANT

Ant jest narzędziem umożliwiającym automatyzację procesów związanych z budowaniem programów. Jego podstawowe cechy to:

- konfiguracja zadań zapisana w formacie XML,
- wieloplatformowość – m. in. Linux, Unix (np. Solaris and HP-UX), Windows 9x i NT, OS/2 Warp, Novell Netware 6 oraz MacOS X.
- rozszerzalność w oparciu o klasy napisane w Javie.

Ant jest rozwijany w ramach Apache Software Foundation. Strona domowa projektu: <http://ant.apache.org>.



# ANT

Przykładowy plik konfiguracyjny dla anta (domyślnie **build.xml**)

```
<project name="project" default="default" basedir=". ">
  <description>
    jakis opis
  </description>

  <target name="default" depends="depends" description="description">
    <jar destfile="bendmetter.jar" includes="**/*.class"
      excludes="tests/*. *" basedir="bin"></jar>
  </target>

  <target name="depends">
  </target>
</project>
```

Aby go “wykonać” wpisujemy w konsoli polecenie **ant**.

DZIĘKUJĘ ZA UWAGĘ