1.

```
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.StringReader;

import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;

public class Main {
   public static void main(String[] args) {
      OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(System.out);
      try {
         XMLReader parser = XMLReaderFactory.createXMLReader();
         InputSource source = new InputSource("rss.xml");
         parser.setContentHandler(new SAXHandler(outputStreamWriter)
         );
         parser.parse(source);
      } catch (Exception e) {
         e.printStackTrace();
      } finally {
         try {
            outputStreamWriter.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
}


class SAXHandler implements ContentHandler { // управление на събитията за
съдържанието на документа
   Locator locator;
   Integer indent;
   OutputStreamWriter outputStreamWriter;
   private final Integer TAB_SIZE = 4;

   public SAXHandler(OutputStreamWriter outputStreamWriter) { // конструктор
      this.outputStreamWriter = outputStreamWriter;
      indent = 0;
   }
```

```
   @Override
   public void setDocumentLocator(Locator locator) { //Позволява на парсера да подате Locator обект на приложението
      this.locator = locator;
   }

   @Override
   public void startDocument() throws SAXException { // Начало на обработката и възникване на първото събитие
      printIndented("<?xml version=\"1.0\" encoding=\"UTF-8\"?>", false, false);
   }

   @Override
   public void endDocument() throws SAXException {
      // ...
   }

   @Override
   public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException { // Достигане до отварящ таг на елемент
      printIndented(String.format("<%s", qName), true, false);
      printAttributes(atts);
      printIndented(">\r\n", false, true);
      ++indent;
   }

   @Override
   public void endElement(String uri, String localName, String qName) throws SAXException { // Достигане до затварящ таг на елемент
      --indent;
      printIndented(String.format("</%s>", qName), true, false);
   }

   @Override
   public void characters(char[] chars, int start, int length) throws SAXException { // Достигане до низ от символи
      String s = new String(chars, start, length).toUpperCase().trim(); // 3/ Текстовите стойности на елементите да бъдат разпечатани с главни букви
      if (s.length() > 0) {
         printIndented(s, false, false);
      }
   }


   @Override
   public void startPrefixMapping(String prefix, String uri) throws SAXException {
      // ...
```

```
    }

    @Override
    public void endPrefixMapping(String prefix) throws SAXException {
        // ...
    }

    @Override
    public void ignorableWhitespace(char[] chars, int start, int length) throws
SAXException {
        // ...
    }

    @Override
    public void processingInstruction(String target, String data) throws
SAXException {
        // ...
    }

    @Override
    public void skippedEntity(String name) throws SAXException {
        // ...
    }

    private void printIndented(String what, boolean isEndOfElement, boolean
isElement) { // принтиране на табулации
        try {
            if(isEndOfElement) { // 2. Всеки елемент да бъде разпечатан на нов
ред:
                outputStreamWriter.write("\r\n");
            }
            if (indent > 0 && !isElement) { // 1. Йерархията на елементите да
бъде запазена
                outputStreamWriter.write(String.format("%1$" + (indent * TAB_SIZE)
+ "s", ""));
            }
            outputStreamWriter.write(what);
            outputStreamWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void printAttributes(Attributes atts) { // 2. Всеки елемент да бъде
разпечатан заедно с включените в него атрибути
        if (atts.getLength() > 0) {
            ++indent;
            for (int i = 0; i < atts.getLength(); ++i) {
```

```
            String name = atts.getQName(i); //връща квалифицираното име на
атрибут на i-та позиция в списъка
            printIndented(String.format(" %s = \"%s\"", name,
atts.getValue(i)), false, true);
         }
         --indent;
      }
   }
}
```

2.

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;

class SAXValidator implements ContentHandler { // управление на събитията за
съдържанието на документа
   Locator locator;
   OutputStreamWriter outputStreamWriter;
   private final Integer TAB_SIZE = 4;
   private String currentElementName;
   private boolean titleDetected = false;
   private int countOfTitle = 0;
   private boolean linkDetected =false;
   private int countOfLink = 0;
   private boolean descriptionDetected = false;
   private int countOfDescription = 0;
   private boolean itemDetected = false;
   private int countOfItem = 0;

   public SAXValidator(OutputStreamWriter outputStreamWriter) {
      this.outputStreamWriter = outputStreamWriter;
   }

   @Override
   public void setDocumentLocator(Locator locator) { // Позволява на парсера
да подате Locator обект на приложението
      this.locator = locator;
   }

   @Override
   public void startDocument() throws SAXException {
```

```
      //..
   }

   @Override
   public void endDocument() throws SAXException {
      // ...
   }

   @Override
   public void startElement(String uri, String localName, String qName,
Attributes atts) throws SAXException { // Достигане до отварящ таг на елемент
      currentElementName = qName; // = квалифицирано име (с префикс) или
празен стринг, ако не се използват такива имена
      validateVersion(atts);

      if (qName.equals("item")) {
         titleDetected = false;
         linkDetected = false;
         descriptionDetected = false;
         countOfTitle = 0;
         countOfDescription = 0;
         countOfLink = 0;
         itemDetected = true;
         ++countOfItem;
      }

      if (qName.equals("title")) {
         titleDetected = true;
         ++countOfTitle;
      }

      if (qName.equals("link")) {
         linkDetected = true;
         ++countOfLink;
      }

      if (qName.equals("description")) {
         descriptionDetected = true;
         ++countOfDescription;
      }
   }

   @Override
   public void endElement(String uri, String localName, String qName) throws
SAXException { // Достигане до затварящ таг на елемент
      if (localName.equals("item")) { // 1. Всеки елемент item трябва да
съдържа едно множество от под-елементите title, link и description, всеки от
тях срещащ се точно един път
```

```
            if (!(titleDetected && linkDetected && descriptionDetected &&
countOfTitle == 1 && countOfLink == 1 && countOfDescription == 1)) {
                reportError("Item must have one subset of the sequence: title,
link, description.");
            }
        }
        if (localName.equals("channel")) { // 3. Елементът channel трябва да
съдържа поне 2 и неповече от 10 под-елемента item
            if (!(itemDetected && countOfItem >= 2 && countOfItem <= 10)) {
                reportError("Number of elements item must be between 2 and 10: ");
            }
        }
    }

    @Override
    public void characters(char[] chars, int start, int length) throws
SAXException {
        //..
    }


    @Override
    public void startPrefixMapping(String prefix, String uri) throws
SAXException {
        // ...
    }

    @Override
    public void endPrefixMapping(String prefix) throws SAXException {
        // ...
    }

    @Override
    public void ignorableWhitespace(char[] chars, int start, int length) throws
SAXException {
        // ...
    }

    @Override
    public void processingInstruction(String target, String data) throws
SAXException {
        // ...
    }

    @Override
    public void skippedEntity(String name) throws SAXException {
        // ...
    }
```

```
    private void printIndented(String what) {
        try {
            outputStreamWriter.write(what);
            outputStreamWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void reportError(String cause) { // 4.
        printIndented(String.format("\r\nError: %s on line %d column %d.",
cause, locator.getLineNumber(), locator.getColumnNumber())); // 4.
Разпечатайте информация за мястото (locator.getLineNumber() = номер на ред и
locator.getColumnNumber() = колона), на което грешката се среща
    }

    private void validateVersion(Attributes atts) { // 2.
        if (atts.getLength() > 0) {
            try {
                if (currentElementName.equals("rss") &&
(Integer.parseInt(atts.getValue("version")) < -1)) { // 2. Стойността на
атрибута version (принадлежащ на елемента rss) трябва да бъде цяло положително
число
                    reportError("Attribute version is expected to have a positive
integer value: ");
                }
            } catch (NumberFormatException e) {
                reportError(String.format("Wrong value for version: %s (Attribute
version is expected to have a positive integer value):",
atts.getValue("version")));
            }
        }
    }
}
```

3.

```
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.StringReader;

import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;
```

```
public class Main {
   public static void main(String[] args) {
      OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(System.out);
      try {
         XMLReader parser = XMLReaderFactory.createXMLReader();
         InputSource source = new InputSource("rss.xml");
         parser.setContentHandler(new SAXTransformator(outputStreamWriter));
         parser.parse(source);
      } catch (Exception e) {
         e.printStackTrace();
      } finally {
         try {
            outputStreamWriter.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
}


class SAXTransformator implements ContentHandler {
   private class Item { // item с под-елементите му title, link и description
      String title;
      String link;
      String description;
   }

   Locator locator;
   OutputStreamWriter outputStreamWriter;
   private final Integer TAB_SIZE = 4;
   private String currentElement;
   private Integer indent;
   private Item currentItem;
   boolean inItem = false;

   public SAXTransformator(OutputStreamWriter outputStreamWriter) {
      this.outputStreamWriter = outputStreamWriter;
   }

   @Override
   public void setDocumentLocator(Locator locator) {
      this.locator = locator;
      indent = 0;
   }

   @Override
```

```
   public void startDocument() throws SAXException { // Начало на обработката
и възникване на първото събитие
      printIndented("<!DOCTYPE html>");
      printIndented("<html>");
      ++indent;
      printIndented("<head><title>List of items</title></head>");
      printIndented("<body>");
      ++indent;
      printIndented("<table style=\"border: 1px solid black;\">");
      ++indent;

printIndented("<thead><tr><th>Title</th><th>Link</th><th>Description</th></tr>
</thead>");
      printIndented("<tbody>");
      ++indent;
   }

   @Override
   public void endDocument() throws SAXException { // Край на бработката и
възникване на последното събитие
      --indent;
      printIndented("</tbody>");
      --indent;
      printIndented("</table>");
      --indent;
      printIndented("</body>");
      --indent;
      printIndented("</html>");
   }

   @Override
   public void startElement(String uri, String localName, String qName,
Attributes atts) throws SAXException { // Достигане до отварящ таг на елемент
      currentElement = qName;

      if ("item".equals(currentElement)) {
         currentItem = new Item();
         inItem = true;
      }
   }

   @Override
   public void endElement(String uri, String localName, String qName) throws
SAXException { //  Достигане до затварящ таг на елемент
      if ("item".equals(localName)) { // <td> = a standard data cell in an
HTML table; <tr> = rows
         printIndented("<tr><td>" + currentItem.title + "</td><td>" +
currentItem.link + "</td><td>" + currentItem.description + "</td></tr>"); //2.
```

```
По един ред за всеки елемент item със стойностите на под-елементите му title,
link и description
         inItem = false;
      }
   }

   @Override
   public void characters(char[] chars, int start, int length) throws
SAXException { //  Достигане до низ от символи
      String s = new String(chars, start, length).trim();

      if (inItem && s.length() > 0) { // 1. Три колони с имена title, link и
description
         if ("title".equals(currentElement)) {
            currentItem.title = s;
         }
         if ("link".equals(currentElement)) {
            currentItem.link = s;
         }
         if ("description".equals(currentElement)) {
            if(currentItem.description == null) {
               currentItem.description = s;
            } else {
               currentItem.description += s;
            }
         }
      }
   }

   @Override
   public void startPrefixMapping(String prefix, String uri) throws
SAXException {
      // ...
   }

   @Override
   public void endPrefixMapping(String prefix) throws SAXException {
      // ...
   }

   @Override
   public void ignorableWhitespace(char[] chars, int start, int length) throws
SAXException {
      // ...
   }

   @Override
   public void processingInstruction(String target, String data) throws
SAXException {
```

```
        // ...
    }

    @Override
    public void skippedEntity(String name) throws SAXException {
        // ...
    }
    private void printIndented(String what) { // метод за принтиране
        try {
            if (indent > 0) {
                outputStreamWriter.write(String.format("%1$" + (indent * TAB_SIZE)
+ "s", ""));
            }
            outputStreamWriter.write(what + "\r\n");
            outputStreamWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

4.

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Attr;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;


public class Main {
    private static boolean skipNL;

    public static void main(String[] args) throws Exception {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setValidating(false);
        DocumentBuilder builder = dbf.newDocumentBuilder();
        InputSource source = new InputSource("rss.xml");
        Document document = builder.parse(source);
        System.out.println(printXML(document.getDocumentElement()));
    }

    private static String printXML(Node rootNode) { // Node - Базов тип данни
на DOM
```

```
     String tab = "    "; // \t
     skipNL = false;
     return(printXML(rootNode, tab));
   }
   private static String printXML(Node rootNode, String tab) {
    String print = "";
    if(rootNode.getNodeType()==Node.ELEMENT_NODE) {
    print += "\n"+tab+"<"+rootNode.getNodeName();
    NamedNodeMap attributes = rootNode.getAttributes(); //NamedNodeMap -
Дефинира неподредено множество от възли, реферирани по име на атрибут
   for (int j = 0; j < attributes.getLength(); j++) {
      Attr attr = (Attr) attributes.item(j); //Attr - Представя атрибут на
елемент
      if(attr != null) { // attr="value"
         print += " " + attr.getNodeName() +  " = \" " + attr.getNodeValue() +
" \" ";
      }
   }
   print += ">";
    }
    NodeList nl = rootNode.getChildNodes(); //  NodeList - Дефинира списък от
възли Nodes на едно ниво в DOM-дървото
    if(nl.getLength()>0) {
      for( int i=0; i<nl.getLength(); i++){
        print += printXML(nl.item(i), tab + " ");
      }
    } else {
     if(rootNode.getNodeValue()!=null) {
      print = rootNode.getNodeValue();
     }
     skipNL = true;
    }
    if(rootNode.getNodeType()==Node.ELEMENT_NODE) {
     if(!skipNL) {
      print += "\n"+tab;
     }
     skipNL = false;
     print += "";
    }
    return(print);
   }
}
```

5.

```
import java.io.File;
```

```java
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

public class Main {
   public static void main(String[] args) throws Exception {
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      dbf.setValidating(false);
      DocumentBuilder builder = dbf.newDocumentBuilder();
      InputSource source = new InputSource("rss.xml");
      Document document = builder.parse(source);
      processTree(document);

      TransformerFactory tf = TransformerFactory.newInstance();
      Transformer writer = tf.newTransformer();
      writer.setOutputProperty(OutputKeys.ENCODING, "utf-8");
      writer.transform(new DOMSource(document), new StreamResult(new
File("rss_new.xml")));
   }
   private static void processTree(Document doc) { // метод за зад.5:
      NodeList linkList = doc.getElementsByTagName("link"); // NodeList =
Дефинира списък от възли Nodes на едно ниво в DOM-дървото
      NodeList itemList = doc.getElementsByTagName("item");
    // 2. Запазете първите 10 item елементи, а всички останали ги изтрийте:
      for (int i = itemList.getLength() - 1; i >= 10; --i) {
         Element item = (Element)itemList.item(i);
         item.getParentNode().removeChild(item); // Методът removeChild -
Изтритият възел се връща като резултат в случай, че е необходима последваща
обработка

      }
    // 1. Превърнете под-елемента link на елемента item в негов атрибут:
      for (int i = linkList.getLength() - 1; i >= 0; --i) { // getLength() -
метод, който намира броя на Nodes в linkList
         Element link = (Element)linkList.item(i);
         Element item = (Element)link.getParentNode();

         if("item".equals(item.getNodeName())) {
            item.setAttribute("link", link.getTextContent().trim()); //
Методът setAttribute - Добавя атрибута link със стойността му към елемента
item
```

```
            item.removeChild(link);
          }
        }
    // 3. Добавете нов под-елемент sponsor на елемента channel:
        Element sponsor = doc.createElement("sponsor"); // createElement -
Метод-фабрика за създаване на възел - нов под-елемент
        sponsor.setTextContent("IBM");
        doc.getElementsByTagName("channel").item(0).appendChild(sponsor);
    }
}
```

Резултат в `rss_new.xml`:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><rss version="0.91">
   <channel company="N.A." location="remote">
       <title>Linux Today</title>
       <link>http://linuxtoday.com</link>
       <language>en-us</language>
       <description>Linux Today News Service</description>
       <image author="anonymous">
           <title>Linux Today</title>
           <url>http://linuxtoday.com/pics/ltnet.png</url>
           <link>http://linuxtoday.com</link>
       </image>
       <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-008-
20-NW-KN-DV">
           <title>CNET News.com: USB 2 arrives in Linux test version</title>

           <description>
               "The USB support in the world of Linux is much more
freewheeling than at
               Microsoft. The Linux USB software has been created by a largely
               self-appointed team of programmers, who feed batches of code to
the
               main kernel project."
           </description>
       </item>
       <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-007-
20-NW-LL">
           <title>
               The Register: Open source developers face new warranty threat
           </title>

           <description>
```

```
                "If there's one thing free software developers hate more than
writing
                documentation, it's fighting a long-drawn out and unglamorous
legal
                battle. But the latest episode in the UCITA saga bodes ill for
any
                free software author based in the United States."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-012-
20-SC-MD">
            <title>Mandrake Linux Security Update Advisory: sudo</title>

            <description>
                "The SuSE Security Team discovered a vulnerability in sudo that
can be
                exploited to obtain root privilege because sudo is installed
setuid
                root. An attacker could trick sudo to log failed sudo calls
                executing the sendmail (or equivalent mailer) program with root
                privileges and an environment that is not completely clean."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-006-
20-PS-BD">
            <title>NewsForge: An Open Source adventure at MacWorld</title>

            <description>
                "'For me, what Darwin brings is not so much another alternative
to Linux
                or FreeBSD or whatever, as those systems I have that are
running
                some Open Source BSD are likely to continue doing that. It's
the
                fact that there is an Open Source kernel and utility suite and
                libraries that will support proprietary commercial applications
that
                I'm willing to pay for but can't run on most of my other
systems.'"
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-005-
20-RV-SW">
            <title>
                Linux Journal: Sysadmin Corner: Unsung Heroes, Part 2
            </title>

            <description>
```

```
                "It seems that several people decided I should show off their
web photo
                album generation tool of choice. So, in response to your
                suggestions, I'm going to put off the cool network tool for
today;
                let's go on the premise that you all took thousands of pictures
over
                the holidays and are dying to make them available on the Web."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-004-
20-SC">
            <title>Conectiva Linux Security Announcement: sudo</title>

            <description>
                "Sebastian Krahmer from SuSe found a vulnerability in the sudo
package which
                could be used by a local attacker to obtain root privileges.
                Versions prior to and including 1.6.3p7 remove a few
potentially
                dangerous environment variables prior to executing a command as
                root, but other variables could be abused and used to obtain
root
                privileges."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-003-
20-PS-KN">
            <title>IBM developerWorks: Introducing XFS</title>

            <description>
                "Up until now, choosing the appropriate next-generation Linux
                filesystem has been refreshingly straightforward. Those who
were
                looking for raw performance generally leaned towards ReiserFS,
while
                those more interested in meticulous data integrity features
                preferred ext3. However, with the release of XFS for Linux,
things
                have suddenly become much more confusing. In particular, it's
no
                longer clear that ReiserFS is still the next-gen performance
                leader."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-002-
20-SC-RH">
            <title>Red Hat Security Advisory: sudo</title>
```

```
            <description>
                "Versions of sudo prior to 1.6.4 would not clear the
environment before
                sending an email notification about unauthorized sudo attempts,
                making it possible for an attacker to supply parameters to the
mail
                program. In the worst case, this could lead to a local root
                exploit."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-16-001-
20-RV-SS">
            <title>
                ZDNet: SuSE 7.3 offers solid server reach and desktop usability
            </title>

            <description>
                "SuSE Linux 7.3 is well prepared for corporate computing,
offering
                support for a broad range of server architectures and
significant
                advances in usability for both administrators and end users.
                Companies looking for a solid server platform--and perhaps even
a
                desktop replacement for Windows--would be well advised to
evaluate
                SuSE 7.3's stellar offerings."
            </description>
        </item>
        <item link="http://linuxtoday.com/news_story.php3?ltsn=2002-01-15-025-
20-NW-KN">
            <title>Linux 2.4.18-pre4 Released</title>

            <description>Changelog, link within.</description>
        </item>




        <textinput>
            <title>Search</title>
            <description>Search Linux Today:</description>
            <name>query</name>
            <link>http://linuxtoday.com/search.php3</link>
        </textinput>
    <sponsor>IBM</sponsor></channel>
</rss>
```

6.

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;

public class Main {
   public static void main(String[] args) {
    XMLInputFactory xmlif = XMLInputFactory.newInstance();
    XMLStreamReader xmlr = null;
      try {
         xmlr = xmlif.createXMLStreamReader(new FileReader("rss.xml"));
       while(xmlr.hasNext()){ // boolean hasNext() – true ако има друго
събитие
       printEvent(xmlr);
       xmlr.next();
       }
       xmlr.close();
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (XMLStreamException e) {
         e.printStackTrace();
      }
    }

   private static void printEvent(XMLStreamReader xmlr) {
      switch (xmlr.getEventType()) {
         case XMLStreamConstants.START_ELEMENT:
          System.out.print("<");
          printName(xmlr);
          printNamespaces(xmlr);
          printAttributes(xmlr);
          System.out.print(">");
          break;
         case XMLStreamConstants.END_ELEMENT:
          System.out.print("</");
          printName(xmlr);
          System.out.print(">");
          break;
         case XMLStreamConstants.SPACE:
         case XMLStreamConstants.CHARACTERS:
          int start = xmlr.getTextStart();
          int length = xmlr.getTextLength();
          System.out.print(new String(xmlr.getTextCharacters(),
```

```
                                start,
                                length));
            break;
           case XMLStreamConstants.PROCESSING_INSTRUCTION:
            System.out.print("<?");
            if (xmlr.hasText())
              System.out.print(xmlr.getText()); //  getText() – връща текстовата
стойност на CHARACTERS, COMMENT, XML ENTITY_REFERENCE, CDATA, SPACE, SAX &
StAX or DTD
            System.out.print("?>");
            break;
           case XMLStreamConstants.CDATA:
            System.out.print("<![CDATA[");
            start = xmlr.getTextStart();
            length = xmlr.getTextLength();
            System.out.print(new String(xmlr.getTextCharacters(),
                                start,
                                length));
            System.out.print("]]>");
            break;
           case XMLStreamConstants.COMMENT:
            System.out.print("<!--");
            if (xmlr.hasText())
              System.out.print(xmlr.getText());
            System.out.print("-->");
            break;
           case XMLStreamConstants.ENTITY_REFERENCE:
            System.out.print(xmlr.getLocalName()+"="); // getLocalName() – взима
името на текущия елемент
            if (xmlr.hasText())
              System.out.print("["+xmlr.getText()+"]");
            break;
           case XMLStreamConstants.START_DOCUMENT:
            System.out.print("<?xml");
            System.out.print(" version='"+xmlr.getVersion()+"'");
            System.out.print("
encoding='"+xmlr.getCharacterEncodingScheme()+"'");
            if (xmlr.isStandalone())
              System.out.print(" standalone='yes'");
            else
              System.out.print(" standalone='no'");
            System.out.print("?>");
            break;
       }
    }

    private static void printName(XMLStreamReader xmlr){ // метод за принтиране
на елемент - вика printName(prefix,uri,localName)
    if(xmlr.hasName()){
```

```
    String prefix = xmlr.getPrefix();
    String uri = xmlr.getNamespaceURI();
    String localName = xmlr.getLocalName(); // getLocalName() – взима името на текущия елемент
    printName(prefix,uri,localName);
   }
   }

   private static void printName(String prefix, String uri, String localName) { // метод за принтиране на елемент
   if (uri != null && !("".equals(uri)) ) System.out.print("['"+uri+"']:");
   if (prefix != null && prefix.length() > 0) System.out.print(prefix+":");
   if (localName != null) System.out.print(localName);
   }

   private static void printAttributes(XMLStreamReader xmlr){ // метод за принтиране на всички атрибути
   for (int i=0; i < xmlr.getAttributeCount(); i++) { // getAttributeCount() – взима броя на атрибутите на текущия елемент
    printAttribute(xmlr,i);
   }
   }

   private static void printAttribute(XMLStreamReader xmlr, int index) { // метод за принтиране на един атрибут
   String prefix = xmlr.getAttributePrefix(index);
   String namespace = xmlr.getAttributeNamespace(index);
   String localName = xmlr.getAttributeLocalName(index); // getAttributeLocalName(index) – името на атрибута
   String value = xmlr.getAttributeValue(index); // getAttributeValue(index) – стойността на атрибута
   System.out.print(" ");
   printName(prefix,namespace,localName);
   System.out.print("='"+value+"'");
   }

   private static void printNamespaces(XMLStreamReader xmlr){ // метод за принтиране на всички пространства от имена
   for (int i=0; i < xmlr.getNamespaceCount(); i++) {
    printNamespace(xmlr,i);
   }
   }

   private static void printNamespace(XMLStreamReader xmlr, int index) { // метод за принтиране на едно пространство от имена
   String prefix = xmlr.getNamespacePrefix(index);
   String uri = xmlr.getNamespaceURI(index);
   System.out.print(" ");
   if (prefix == null)
```

```
    System.out.print("xmlns='"+uri+"'");
   else
    System.out.print("xmlns:"+prefix+"='"+uri+"'");
   }
}
```

7.

```
import java.io.IOException;
import java.io.StringWriter;
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamWriter;

public class Main {
   public static void main(String[] args) {
       try {
           StringWriter stringWriter = new StringWriter();
           XMLOutputFactory xMLOutputFactory = XMLOutputFactory.newInstance();
           XMLStreamWriter xMLStreamWriter =
xMLOutputFactory.createXMLStreamWriter(stringWriter);
           xMLStreamWriter.writeStartDocument(); //writeStartDocument(version)
– записва XML хедър (оглавление)
           xMLStreamWriter.writeStartElement("bookstore"); //
writeStartElement(name) – записва стартов маркер
           xMLStreamWriter.writeStartElement("book");//book 1
           xMLStreamWriter.writeAttribute("category", "COOKING");
//writeAttribute(name, value) – записва атрибут category="COOKING"
           createSimpleElement(xMLStreamWriter, "title", "lang", "en",
"Everyday Italian");
           createSimpleElement(xMLStreamWriter, "author", null, null, "Giada De
Laurentiis");
           createSimpleElement(xMLStreamWriter, "year", null, null, "2005");
           createSimpleElement(xMLStreamWriter, "price", null, null, "30.00");
           xMLStreamWriter.writeEndElement();//book 1
           xMLStreamWriter.writeStartElement("book");//book 2
           xMLStreamWriter.writeAttribute("category", "CHILDREN"); //
//writeAttribute(name, value) – записва атрибут category="CHILDREN"
           createSimpleElement(xMLStreamWriter, "title", "lang", "en", "Harry
Potter");
           createSimpleElement(xMLStreamWriter, "author", null, null, "J K.
Rowling");
           createSimpleElement(xMLStreamWriter, "year", null, null, "2005");
           createSimpleElement(xMLStreamWriter, "price", null, null, "29.99");
           xMLStreamWriter.writeEndElement();//book 2
           xMLStreamWriter.writeStartElement("book"); //book 3
```

```
            xMLStreamWriter.writeAttribute("category", "WEB"); //
//writeAttribute(name, value) – записва атрибут category="WEB"
            createSimpleElement(xMLStreamWriter, "title", "lang", "en",
"Learning XML");
            createSimpleElement(xMLStreamWriter, "author", null, null, "Erik T.
Ray");
            createSimpleElement(xMLStreamWriter, "year", null, null, "2003");
            createSimpleElement(xMLStreamWriter, "price", null, null, "39.95");
            xMLStreamWriter.writeEndElement();//book 3
            xMLStreamWriter.writeEndDocument();
            xMLStreamWriter.flush(); //↑ flush() – прави запис на буферирания
изход
            xMLStreamWriter.close(); // //close() – затваря XMLStreamWriter
            String xmlString = stringWriter.getBuffer().toString();
            stringWriter.close(); //close() – затваря stringWriter
            System.out.println(xmlString);
         } catch (XMLStreamException e) {
            e.printStackTrace();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }

   public static void createSimpleElement(XMLStreamWriter xMLStreamWriter,
String elementName, String attributeName, String attributeValue, String
strValue) {
         try {
            if(xMLStreamWriter != null && elementName != null) {
               xMLStreamWriter.writeStartElement(elementName); //
writeStartElement(name) – записва стартов маркер
               if(attributeName != null) {
                  xMLStreamWriter.writeAttribute(attributeName, attributeValue);
// writeAttribute(name, value) – записва атрибут
               }
               if(strValue != null) {
                  xMLStreamWriter.writeCharacters(strValue); //
writeCharacters(value) – записва текст, кодирайки символи като <, > и &
               }
               xMLStreamWriter.writeEndElement(); // writeEndElement() – записва
краен маркер
            }
         } catch (XMLStreamException e) {
            e.printStackTrace();
         }
   }
}
```