```java
//ItemListParser.java

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

/**
   An XML parser for item lists
*/
public class ItemListParser
{
   /**      Constructs a parser that can parse item lists    */
      private DocumentBuilder builder;
      private XPath path;

   public ItemListParser()
         throws ParserConfigurationException
   {
      DocumentBuilderFactory dbfactory
            = DocumentBuilderFactory.newInstance();
      builder = dbfactory.newDocumentBuilder();
      XPathFactory xpfactory = XPathFactory.newInstance();
      path = xpfactory.newXPath();
   }

   /**
      Parses an XML file containing an item list
      @param fileName the name of the file
      @return an array list containing all items in the XML file
   */
   public ArrayList<LineItem> parse(String fileName)
      throws SAXException, IOException, XPathExpressionException
   {
      File f = new File(fileName);
      Document doc = builder.parse(f);

      ArrayList<LineItem> items = new ArrayList<LineItem>();
      int itemCount = Integer.parseInt(path.evaluate(
            "count(/items/item)", doc));
      for (int i = 1; i <= itemCount; i++)
      {
         String description = path.evaluate(
               "/items/item[" + i + "]/product/description", doc);
         double price = Double.parseDouble(path.evaluate(
               "/items/item[" + i + "]/product/price", doc));
         Product pr = new Product(description, price);
         int quantity = Integer.parseInt(path.evaluate(
               "/items/item[" + i + "]/quantity", doc));
         LineItem it = new LineItem(pr, quantity);
         items.add(it);
      }
      return items;
   }

}
```

```java
import java.util.ArrayList;
public class MyDomParser
{  public static void main (String [] args) throws Exception
   {  ItemListParser parser = new ItemListParser();
      ArrayList <LineItem> items = parser.parse("items.xml");
      for( LineItem anItem : items)
            System.out.println( anItem.format());
   }
}
```

```java
//MyDomParser.java

import java.util.ArrayList;

/**
   This program parses an XML file containing an item list.
   It prints out the items that are described in the XML file.
*/
public class MyDomParser
{
   public static void main(String[] args) throws Exception
   {
      ItemListParser parser = new ItemListParser();
      ArrayList<LineItem> items = parser.parse("items.xml");
      for (LineItem anItem : items)
         System.out.println(anItem.format());
   }
}
```

```java
//Product.java

/**
   Describes a product with a description and a price
*/
class Product
{
   private String description;
   private double price;

/**
      Constructs a product from a description and a price.
      @param aDescription the product description
      @param aPrice the product price
   */
   public Product(String aDescription, double aPrice)
   {
      description = aDescription;
      price = aPrice;
   }

   /**
      Gets the product description.
      @return the description
   */
   public String getDescription()
   {
      return description;
   }

   /**
      Gets the product price.
      @return the unit price
   */
   public double getPrice()
   {
      return price;
   }

}
```

```
//items.xml
<?xml version="1.0"?>
<items>
   <item>
      <product>
         <description>Ink Jet Refill Kit</description>
         <price>29.95</price>
      </product>
      <quantity>8</quantity>
   </item>
   <item>
      <product>
         <description>4-port Mini Hub</description>
         <price>19.95</price>
      </product>
      <quantity>4</quantity>
   </item>
</items>

//LineItem.java
/**
   Describes a quantity of an article to purchase.
*/
public class LineItem
{
      private int quantity;
      private Product theProduct;

   /**
      Constructs an item from the product and quantity.
      @param aProduct the product
      @param aQuantity the item quantity
   */
   public LineItem(Product aProduct, int aQuantity)
   {
      theProduct = aProduct;
      quantity = aQuantity;
   }

   /**
      Computes the total cost of this line item.
      @return the total price
   */
   public double getTotalPrice()
   {
      return theProduct.getPrice() * quantity;
   }

   /**
      Formats this item.
      @return a formatted string of this item
   */
   public String format()
   {
      return String.format("%-30s%8.2f%5d%8.2f",
            theProduct.getDescription(), theProduct.getPrice(),
            quantity, getTotalPrice());
   }


}
```