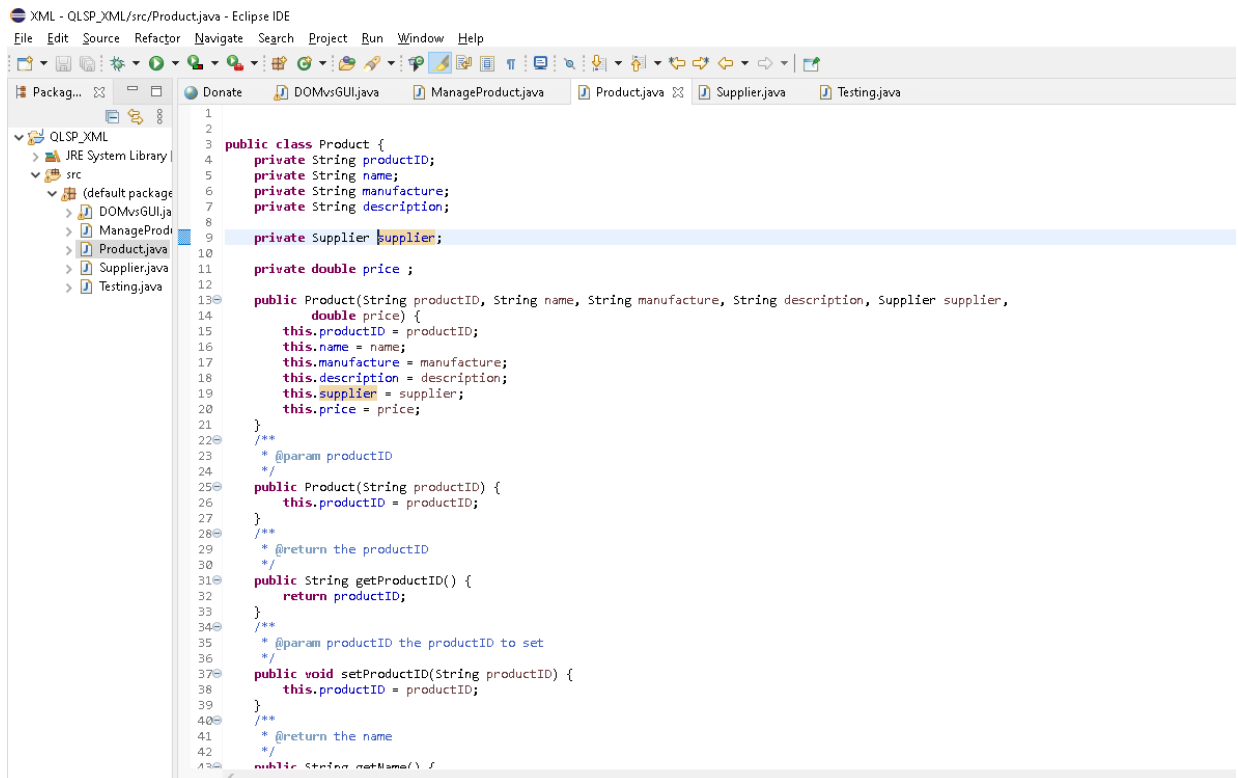
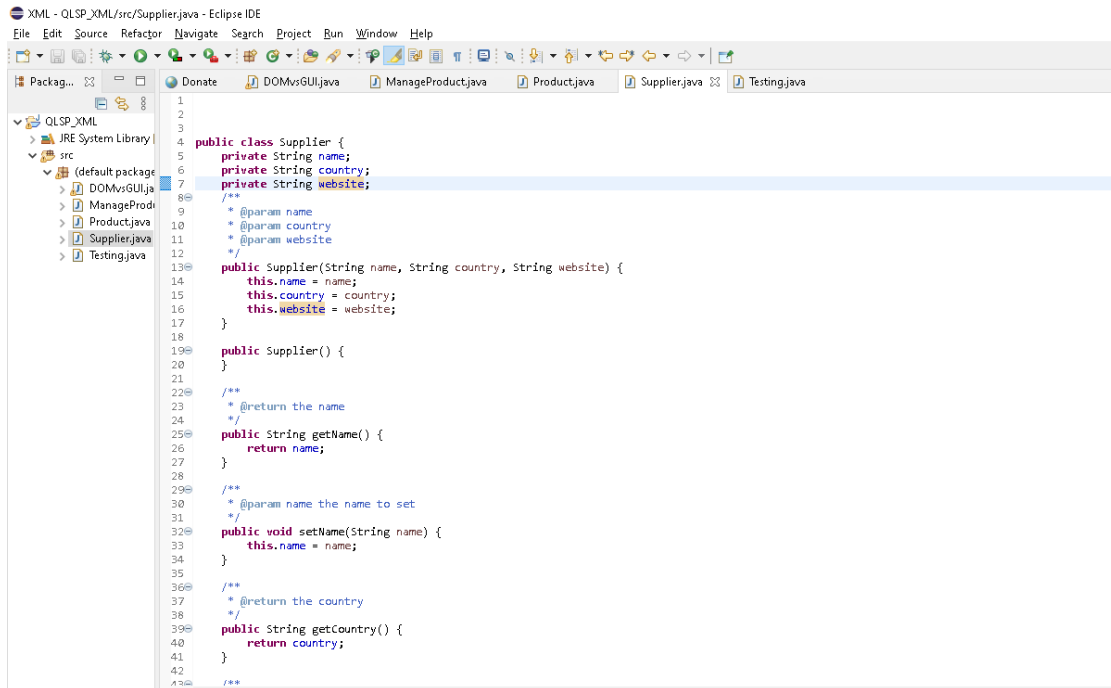


# Bài thực hành XML



The screenshot shows the Eclipse IDE with the file 'Product.java' open. The package explorer on the left shows the project structure: 'QLSP\_XML' containing 'src' with sub-packages 'DOMvsGUILjava', 'ManageProductjava', 'Productjava', 'Supplierjava', and 'Testingjava'. The 'Product.java' file is selected, and its code is displayed in the editor. The code defines a 'Product' class with private attributes 'productID', 'name', 'manufacture', 'description', 'supplier', and 'price'. It includes a constructor and several getter and setter methods. The 'supplier' attribute is highlighted in blue in the code.

```
1
2
3 public class Product {
4     private String productID;
5     private String name;
6     private String manufacture;
7     private String description;
8
9     private Supplier supplier;
10
11     private double price ;
12
13     public Product(String productID, String name, String manufacture, String description, Supplier supplier,
14                     double price) {
15         this.productID = productID;
16         this.name = name;
17         this.manufacture = manufacture;
18         this.description = description;
19         this.supplier = supplier;
20         this.price = price;
21     }
22     /**
23      * @param productID
24      */
25     public Product(String productID) {
26         this.productID = productID;
27     }
28     /**
29      * @return the productID
30      */
31     public String getProductID() {
32         return productID;
33     }
34     /**
35      * @param productID the productID to set
36      */
37     public void setProductID(String productID) {
38         this.productID = productID;
39     }
40     /**
41      * @return the name
42      */
43     public String getName() {
```



The screenshot shows the Eclipse IDE with the file 'Supplier.java' open. The package explorer on the left shows the project structure: 'QLSP\_XML' containing 'src' with sub-packages 'DOMvsGUILjava', 'ManageProductjava', 'Productjava', 'Supplierjava', and 'Testingjava'. The 'Supplier.java' file is selected, and its code is displayed in the editor. The code defines a 'Supplier' class with private attributes 'name', 'country', and 'website'. It includes a constructor, a default constructor, and several getter and setter methods. The 'website' attribute is highlighted in blue in the code.

```
1
2
3 public class Supplier {
4     private String name;
5     private String country;
6     private String website;
7
8     /**
9      * @param name
10     * @param country
11     * @param website
12     */
13     public Supplier(String name, String country, String website) {
14         this.name = name;
15         this.country = country;
16         this.website = website;
17     }
18
19     public Supplier() {
20     }
21
22     /**
23      * @return the name
24      */
25     public String getName() {
26         return name;
27     }
28
29     /**
30      * @param name the name to set
31      */
32     public void setName(String name) {
33         this.name = name;
34     }
35
36     /**
37      * @return the country
38      */
39     public String getCountry() {
40         return country;
41     }
42
43     /**
```

```
File Edit Source Refactor Navigate Search Project Run Window Help
Packag... QLSP_XML JRE System Library src
  (default package)
  DOMvsgUI.java
  ManageProduct.java
  Product.java
  Supplier.java
  Testing.java

21 import java.io.IOException;
22
23 public class ManageProduct {
24     private static String fileName = "src/qlsp/products.xml";
25
26     private DocumentBuilderFactory factory;
27     private DocumentBuilder builder;
28     private Document document;
29
30     public ManageProduct() {
31         try {
32             factory = DocumentBuilderFactory.newInstance();
33             builder = factory.newDocumentBuilder();
34             document = builder.parse(fileName);
35         } catch (ParserConfigurationException e) {
36             e.printStackTrace();
37         } catch (SAXException e) {
38             e.printStackTrace();
39         } catch (IOException e) {
40             e.printStackTrace();
41         }
42     }
43
44     public ArrayList<Product> getAllProducts()
45     {
46         ArrayList<Product> list = new ArrayList<Product>();
47
48         Element root = document.getDocumentElement();
49         NodeList plist = root.getElementsByTagName("product");
50
51         int pCount = plist.getLength();
52         for (int i = 0; i < pCount; i++) {
53             Element pNode = (Element) plist.item(i);
54             String productID = pNode.getAttribute("id");
55             String name = pNode.getElementsByTagName("productName").item(0).getTextContent();
56             String manufacture = pNode.getElementsByTagName("manufacture").item(0).getTextContent();
57             String description = pNode.getElementsByTagName("description").item(0).getTextContent();
58
59             Element sNode = (Element) pNode.getElementsByTagName("supplier").item(0);
60             String sName = sNode.getElementsByTagName("supplierName").item(0).getTextContent();
61             String country = sNode.getElementsByTagName("country").item(0).getTextContent();
62             String website = sNode.getElementsByTagName("website").item(0).getTextContent();
```

```
61 String website = sNode.getElementsByTagName("website").item(0).getTextContent();
62
63 Supplier supplier = new Supplier(sName, country, website);
64
65 String sprice = pNode.getElementsByTagName("price").item(0).getTextContent();
66 double price = 0.0;
67 try {
68     price = Double.valueOf(sprice);
69 } catch (NumberFormatException ex) {}
70
71 Product p = new Product(productID, name, manufacture, description, supplier, price);
72 list.add(p);
73 }
74
75 return list;
76
77 /**
78  * Add
79  * @param p
80  */
81 public void addProduct(Product p) {
82     Element root = document.getDocumentElement();
83
84     Element pNode = document.createElement("product");
85     root.appendChild(pNode);
86     pNode.setAttribute("id", p.getProductID());
87
88     Node nameNode0 = document.createElement("productName");
89     pNode.appendChild(nameNode0);
90     nameNode0.setTextContent(p.getName());
91
92     Node nameNode1 = document.createElement("manufacture");
93     pNode.appendChild(nameNode1);
94     nameNode1.setTextContent(p.getManufacture());
95
96     Node nameNode2 = document.createElement("description");
97     pNode.appendChild(nameNode2);
98     nameNode2.setTextContent(p.getManufacture());
99
100     Element pNodes = document.createElement("supplier");
101     pNode.appendChild(pNodes);
102     Node nameNode11 = document.createElement("supplierName");
```

```
100 pNode.appendChild(pNodes);
101 Node nameNode11 = document.createElement("supplierName");
102 pNodes.appendChild(nameNode11);
103 nameNode11.setTextContent(p.getSupplier().getName());
104
105 Node nameNode22 = document.createElement("country");
106 pNodes.appendChild(nameNode22);
107 nameNode22.setTextContent(p.getSupplier().getCountry());
108
109 Node nameNode33 = document.createElement("website");
110 pNodes.appendChild(nameNode33);
111 nameNode33.setTextContent(p.getSupplier().getWebsite());
112 Node nameNode3 = document.createElement("price");
113 pNode.appendChild(nameNode3);
114 nameNode3.setTextContent(p.getPrice()+"");
115
116 }
117 /**
118  * Delete
119  * @param pid
120  */
121 public void deleteProduct(String pid) {
122     Element root = document.getDocumentElement();
123     NodeList pList = root.getElementsByTagName("product");
124     for (int i = 0; i < pList.getLength(); i++) {
125         Element pNode = (Element) pList.item(i);
126         String productID = pNode.getAttribute("id");
127         if (productID.equalsIgnoreCase(pid)) {
128             pNode.getParentNode().removeChild(pNode);
129             break;
130         }
131     }
132 }
133
134 /**
135  * Update
136  * @param pid
137  * @param newPrice
138  */
139 public void updatePrice(String pid, double newPrice) {
140     Element root = document.getDocumentElement();
141     NodeList pList = root.getElementsByTagName("product");
142     for (int i = 0; i < pList.getLength(); i++) {
143         Element pNode = (Element) pList.item(i);
144         String productID = pNode.getAttribute("id");
145         if (productID.equalsIgnoreCase(pid)) {
146             Node priceNode = pNode.getElementsByTagName("price").item(0);
147             priceNode.setTextContent(newPrice + "");
148             break;
149         }
150     }
151 }
152
153 /**
154  * Write XML file
155  */
156 public void writeXMLFile() {
157     TransformerFactory factory = null;
158     Transformer transformer = null;
159     try {
160         factory = TransformerFactory.newInstance();
161         transformer = factory.newTransformer();
162         transformer.setOutputProperty(OutputKeys.INDENT, "yes");
163         transformer.transform(new DOMSource(document), new StreamResult(fileName));
164     } catch (Exception ex) {
165         ex.printStackTrace();
166     }
167 }
168
169 /**
170  * Console
171  */
172 public void printAll() {
173     TransformerFactory factory = null;
174     Transformer transformer = null;
175     try {
176         factory = TransformerFactory.newInstance();
177         transformer = factory.newTransformer();
178         transformer.setOutputProperty(OutputKeys.INDENT, "yes");
179         transformer.transform(new DOMSource(document), new StreamResult(System.out));
180     } catch (Exception ex) {
181         ex.printStackTrace();
182     }
183 }
```

```
139
140 public void updatePrice(String pid, double newPrice) {
141     Element root = document.getDocumentElement();
142     NodeList pList = root.getElementsByTagName("product");
143     for (int i = 0; i < pList.getLength(); i++) {
144         Element pNode = (Element) pList.item(i);
145         String productID = pNode.getAttribute("id");
146         if (productID.equalsIgnoreCase(pid)) {
147             Node priceNode = pNode.getElementsByTagName("price").item(0);
148             priceNode.setTextContent(newPrice + "");
149             break;
150         }
151     }
152 }
153
154 /**
155  * Write XML file
156  */
157 public void writeXMLFile() {
158     TransformerFactory factory = null;
159     Transformer transformer = null;
160     try {
161         factory = TransformerFactory.newInstance();
162         transformer = factory.newTransformer();
163         transformer.setOutputProperty(OutputKeys.INDENT, "yes");
164         transformer.transform(new DOMSource(document), new StreamResult(fileName));
165     } catch (Exception ex) {
166         ex.printStackTrace();
167     }
168 }
169
170 /**
171  * Console
172  */
173 public void printAll() {
174     TransformerFactory factory = null;
175     Transformer transformer = null;
176     try {
177         factory = TransformerFactory.newInstance();
178         transformer = factory.newTransformer();
179         transformer.setOutputProperty(OutputKeys.INDENT, "yes");
180         transformer.transform(new DOMSource(document), new StreamResult(System.out));
181     } catch (Exception ex) {
182         ex.printStackTrace();
183     }
184 }
```

```

171     * Console
172     */
173     public void printAll() {
174         TransformerFactory factory = null;
175         Transformer transformer = null;
176         try {
177             factory = TransformerFactory.newInstance();
178             transformer = factory.newTransformer();
179             transformer.setOutputProperty(OutputKeys.INDENT, "yes");
180             transformer.transform(new DOMSource(document), new StreamResult(System.out));
181         } catch (Exception ex) {
182             ex.printStackTrace();
183         }
184     }
185 }
186 }
187

```

XML - QLSP\_XML/src/testing.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

```

1  import java.util.Scanner;
2  public class Testing {
3      private static Scanner sc = new Scanner(System.in);
4
5      public static void main(String[] args) {
6          int luaChon;
7
8          ManageProduct dom = new ManageProduct();
9
10         boolean flag = true;
11
12         do {
13             createMenu();
14             luaChon = sc.nextInt();
15             switch (luaChon) {
16                 case 1:
17                     sc.nextLine();
18                     Product p = createNewProduct();
19                     dom.addProduct(p);
20                     break;
21                 case 2:
22                     sc.nextLine();
23                     System.out.println("Enter productID: ");
24                     String productID = sc.nextLine();
25                     dom.deleteProduct(productID);
26                     break;
27                 case 3:
28                     sc.nextLine();
29                     System.out.println("Enter productID: ");
30                     productID = sc.nextLine();
31                     System.out.println("Enter new price: ");
32                     double donGia = sc.nextDouble();
33                     dom.updatePrice(productID, donGia);
34                     break;
35                 case 4:
36                     dom.printAll();
37                     break;
38                 case 5:
39                     dom.writeXMLFile();
40                     break;
41             }
42         } while (flag);
43     }
44 }

```

```

40         case 5:
41             dom.writeXMLFile();
42             break;
43         default:
44             flag = false;
45             break;
46     }
47     } while (flag);
48 }
49
50 private static Product createNewProduct() {
51     System.out.println("Enter productID: ");
52     String productID = sc.nextLine();
53     System.out.println("Enter name: ");
54     String name = sc.nextLine();
55     System.out.println("Enter manufacture: ");
56     String manufacture = sc.nextLine();
57     System.out.println("Enter description: ");
58     String description = sc.nextLine();
59     Supplier supplier = createNewSupplier();
60     System.out.println("Enter price: ");
61     double price = sc.nextDouble();
62
63     return new Product(productID, name, manufacture, description, supplier, price);
64 }
65
66 private static Supplier createNewSupplier() {
67     System.out.println("Enter supplier name: ");
68     String name = sc.nextLine();
69     System.out.println("Enter country: ");
70     String country = sc.nextLine();
71     System.out.println("Enter website: ");
72     String website = sc.nextLine();
73     return new Supplier(name, country, website);
74 }
75
76 private static void createMenu() {
77     System.out.println("=====Menu=====");
78     System.out.println("1. Add product");
79     System.out.println("2. Delete product");
80     System.out.println("3. Update price");
81     System.out.println("4. Print all");
82 }

```

```
76
77 private static void createMenu() {
78     System.out.println("====Menu====");
79     System.out.println("1. Add product");
80     System.out.println("2. Delete product");
81     System.out.println("3. Update price");
82     System.out.println("4. Print all");
83     System.out.println("5. Write XML file");
84     System.out.println("0. Exit");
85     System.out.print("Your choice: ");
86 }
87 }
```