**ASSIGNMENT**

**package** accolite;

**import** java.util.\* ;

**import** java.text.SimpleDateFormat;

**import** java.io.\*;

**import** javax.xml.parsers.DocumentBuilderFactory;

**import** javax.xml.transform.\*;

**import** javax.xml.transform.dom.DOMSource;

**import** javax.xml.transform.stream.StreamResult;

**import** javax.xml.parsers.DocumentBuilder;

**import** org.w3c.dom.Document;

**import** org.w3c.dom.NodeList;

**import** org.w3c.dom.Node;

**import** org.w3c.dom.Element;

**public** **class** Annotations {

**public** **static** **void** main(String[] args) {

**try** {

File inputFile1 = **new** File("E:/SAU-2021-Jan-Batch-Delhi/Java-Evening/file1.txt");

File inputFile2 = **new** File("E:/SAU-2021-Jan-Batch-Delhi/Java-Evening/file2.txt");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc1 = dBuilder.parse(inputFile1);

Document doc2 = dBuilder.parse(inputFile2);

Document merged\_doc = dBuilder.newDocument() ;

Document validDoc = dBuilder.newDocument() ;

Document invalidDoc = dBuilder.newDocument() ;

doc1.getDocumentElement().normalize();

doc2.getDocumentElement().normalize();

NodeList doc1\_CSR = doc1.getElementsByTagName("CSR\_Producer");

NodeList doc2\_CSR = doc2.getElementsByTagName("CSR\_Producer");

Element root = merged\_doc.createElement("CSR\_Producer");

merged\_doc.appendChild(root);

System.***out***.println();

**for**(**int** temp = 0; temp < doc1\_CSR.getLength(); temp++) {

Node curr\_CSR1 = doc1\_CSR.item(temp);

**for**(**int** i = 0 ; i < doc2\_CSR.getLength() ; i++){

Node curr\_CSR2 = doc2\_CSR.item(i) ;

Element curr\_Ele1 = (Element) curr\_CSR1 ;

Element curr\_Ele2 = (Element) curr\_CSR2 ;

**if**( curr\_Ele1.getAttribute("NIPR\_Number").equals(curr\_Ele2.getAttribute("NIPR\_Number")) ){

NodeList arr1 = curr\_Ele1.getElementsByTagName("License");

**for**(**int** j = 0 ; j < arr1.getLength() ; j++){

Element license\_ele1 = (Element)arr1.item(j);

NodeList arr2 = curr\_Ele2.getElementsByTagName("License");

**for**(**int** k = 0 ; k < arr2.getLength() ; k++){

Element license\_ele2 = (Element)arr2.item(k) ;

**if**(license\_ele1.getAttribute("State\_Code").equals(license\_ele2.getAttribute("State\_Code"))

&& license\_ele1.getAttribute("License\_Number").equals(license\_ele2.getAttribute("License\_Number"))

&& license\_ele1.getAttribute("License\_Expiration\_Date").equals(license\_ele2.getAttribute("License\_Expiration\_Date"))){

Node newEle = merged\_doc.importNode(license\_ele2, **true**) ;

root.appendChild(newEle) ;

System.***out***.println("After Copy...");

*prettyPrint*(merged\_doc,"E:/SAU-2021-Jan-Batch-Delhi/Java-Evening/merged\_file.txt");

}

}

}

}

}

}

Element validRoot = validDoc.createElement("CSR\_Producer");

validDoc.appendChild(validRoot);

Element invalidRoot = invalidDoc.createElement("CSR\_Producer");

invalidDoc.appendChild(invalidRoot);

NodeList mergedDoc\_CSR = merged\_doc.getElementsByTagName("CSR\_Producer");

**for**(**int** i = 0 ; i < mergedDoc\_CSR.getLength() ; i++){

Element curr\_ele = (Element)mergedDoc\_CSR.item(i) ;

NodeList license\_list = curr\_ele.getElementsByTagName("License") ;

**for**(**int** j = 0 ; j < license\_list.getLength() ; j++){

Element curr\_license = (Element)license\_list.item(j) ;

String date = curr\_license.getAttribute("License\_Expiration\_Date");

Date expiry = **new** SimpleDateFormat("dd/MM/yyyy").parse(date);

Date curr = **new** Date() ;

**if**(expiry.compareTo(curr) > 0){

Node newEle = validDoc.importNode(curr\_license, **true**) ;

validRoot.appendChild(newEle) ;

*prettyPrint*(validDoc,"E:/SAU-2021-Jan-Batch-Delhi/Java-Evening/ValidEntries.txt");

}**else**{

Node newEle = invalidDoc.importNode(curr\_license, **true**) ;

invalidRoot.appendChild(newEle) ;

*prettyPrint*(invalidDoc,"E:/SAU-2021-Jan-Batch-Delhi/Java-Evening/InvalidEntries.txt");

}

}

}

}

**catch**(Exception e){

e.printStackTrace();

System.***out***.println() ;

}

}

**public** **static** **final** **void** prettyPrint(Document xml , String filename) **throws** Exception {

Transformer tf = TransformerFactory.*newInstance*().newTransformer();

tf.transform(**new** DOMSource(xml), **new** StreamResult(**new** File(filename)));

}

}