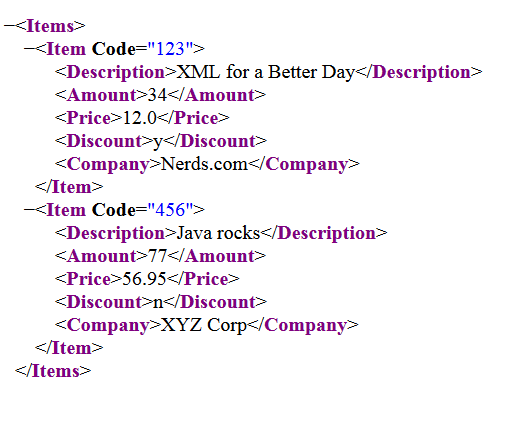
**CSC 238 Session 9**

**Introduction to XML**

**====================================================================**

**Items.xml**

**Item.java**

**import java.text.NumberFormat;**

**public class Item {**

**private String code;**

**private String description;**

**private int amount;**

**private double price;**

**private char discount;**

**private String company;**

**public Item() {**

**this("", "", 0, 0.0, ' ',"");**

**}**

**public Item( String code, String description, int amount, double price, char discount,String company) {**

**this.code = code;**

**this.description = description;**

**this.amount = amount;**

**this.price = price;**

**this.discount = discount;**

**this.company = company;**

**}**

**public void setCode(String code) {**

**this.code = code;**

**}**

**public String getCode() {**

**return code;**

**}**

**public void setDescription(String description) {**

**this.description = description;**

**}**

**public String getDescription() {**

**return description;**

**}**

**public void setAmount(int amount) {**

**this.amount = amount;**

**}**

**public double getAmount() {**

**return amount;**

**}**

**public String getAmountString() {**

**String amountString = Integer.toString(amount);**

**return amountString;**

**}**

**public void setPrice(double price) {**

**this.price = price;**

**}**

**public double getPrice() {**

**return price;**

**}**

**public String getFormattedPrice() {**

**NumberFormat currency = NumberFormat.getCurrencyInstance();**

**return currency.format(price);**

**}**

**public void setDiscount(char discount) {**

**this.discount = discount;**

**}**

**public char getDiscount() {**

**return discount;**

**}**

**public void setCompany(String company) {**

**this.company = company;**

**}**

**public String getCompany() {**

**return company;**

**}**

**public boolean equals(Object object) {**

**if (object instanceof Item) {**

**Item item2 = (Item) object;**

**if ( code.equals(item2.getCode())**

**&& description.equals(item2.getDescription())**

**&& amount == item2.getAmount()**

**&& price == item2.getPrice()**

**&& discount == item2.getDiscount()**

**&& company.equals(item2.getCompany())**

**) {**

**return true;**

**}**

**}**

**return false;**

**}**

**public String toString() {**

**return "Code: " + code + "\n"**

**+ "Description: " + description + "\n"**

**+ "amount: " + amount + "\n"**

**+ "Price: " + this.getFormattedPrice() + "\n"**

**+ "discount: " + discount + "\n"**

**+ "company is " + company;**

**}**

**}**

**//======================================================================================**

**//ItemReader.java**

**import java.util.ArrayList;**

**public interface ItemReader**

**{**

**Item getItem(String code);**

**ArrayList<Item> getItems();**

**}**

**//============================================================================================**

**//ItemWriter.java**

**public interface ItemWriter**

**{**

**boolean addItem(Item p);**

**boolean updateItem(Item p);**

**boolean deleteItem(Item p);**

**}**

**//==========================================================================================**

**// ItemDAO.java**

**public interface ItemDAO extends ItemReader, ItemWriter, ItemConstants**

**{**

**// all methods from the ItemReader and ItemWriter interfaces**

**// all static constants from the ItemConstants interface**

**}**

**//=============================================================================================**

**//ItemMaintApp.java**

**import java.util.Scanner;**

**import java.util.ArrayList;**

**public class ItemMaintApp implements ItemConstants {**

**// declare two class variables**

**private static ItemDAO itemDAO = null;**

**private static Scanner sc = null;**

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

**// display a welcome message**

**System.out.println("Welcome to the Item Maintenance application\n");**

**// set the class variables**

**itemDAO = DAOFactory.getItemDAO();**

**sc = new Scanner(System.in);**

**// display the command menu**

**displayMenu();**

**// perform 1 or more actions**

**String action = "";**

**while (!action.equalsIgnoreCase("exit")) {**

**// get the input from the user**

**action = Validator.getString(sc,**

**"Enter a command: ");**

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

**if (action.equalsIgnoreCase("list")) {**

**displayAllItems();**

**} else if (action.equalsIgnoreCase("add")) {**

**addItem();**

**} else if (action.equalsIgnoreCase("del") || action.equalsIgnoreCase("delete")) {**

**deleteItem();**

**} else if (action.equalsIgnoreCase("update")) {**

**updateItem();**

**} else if (action.equalsIgnoreCase("help") || action.equalsIgnoreCase("menu")) {**

**displayMenu();**

**} else if (action.equalsIgnoreCase("exit")) {**

**System.out.println("Bye.\n");**

**} else {**

**System.out.println("Error! Not a valid command.\n");**

**}**

**}**

**}**

**public static void displayMenu() {**

**System.out.println("COMMAND MENU");**

**System.out.println("list - List all items");**

**System.out.println("add - Add a item");**

**System.out.println("del - Delete a item");**

**System.out.println("update - update a item");**

**System.out.println("help - Show this menu");**

**System.out.println("exit - Exit this application\n");**

**}**

**public static void displayAllItems() {**

**System.out.println("ITEM LIST");**

**ArrayList<Item> items = itemDAO.getItems();**

**if (items == null) {**

**System.out.println("Error! Unable to get items.\n");**

**} else {**

**Item p = null;**

**StringBuilder sb = new StringBuilder();**

**for (int i = 0; i < items.size(); i++) {**

**p = items.get(i);**

**sb.append(**

**StringUtils.padWithSpaces(**

**p.getCode(), CODE\_SIZE + 4)**

**+ StringUtils.padWithSpaces(**

**p.getDescription(), DESCRIPTION\_SIZE + 4)**

**+ StringUtils.padWithSpaces(**

**p.getAmountString(), CODE\_SIZE + 4)**

**+ p.getFormattedPrice() + "\n"**

**);**

**}**

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

**}**

**}**

**public static void addItem() {**

**String code = Validator.getString(**

**sc, "Enter item code: ");**

**String description = Validator.getLine(**

**sc, "Enter item description: ");**

**int amount = Validator.getInt(**

**sc, "Enter item amount: ");**

**double price = Validator.getDouble(**

**sc, "Enter price: ");**

**char discount = Validator.getChar(**

**sc, "Enter does the item has discount: ");**

**String company = Validator.getLine(**

**sc, "Enter the company: ");**

**Item item = new Item();**

**item.setCode(code);**

**item.setDescription(description);**

**item.setAmount(amount);**

**item.setPrice(price);**

**item.setDiscount(discount);**

**item.setCompany(company);**

**boolean success = itemDAO.addItem(item);**

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

**if (success) {**

**System.out.println(description**

**+ " was added to the database.\n");**

**} else {**

**System.out.println("Error! Unable to add item\n");**

**}**

**}**

**public static void deleteItem() {**

**String code = Validator.getString(sc,**

**"Enter item code to delete: ");**

**Item p = itemDAO.getItem(code);**

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

**if (p != null) {**

**boolean success = itemDAO.deleteItem(p);**

**if (success) {**

**System.out.println(p.getDescription()**

**+ " was deleted from the database.\n");**

**} else {**

**System.out.println("Error! Unable to add item\n");**

**}**

**} else {**

**System.out.println("No item matches that code.\n");**

**}**

**}**

**public static void updateItem() {**

**String code = Validator.getString(**

**sc, "Enter item code: ");**

**String description = Validator.getLine(**

**sc, "Enter item description: ");**

**int amount = Validator.getInt(**

**sc, "Enter item amount: ");**

**double price = Validator.getDouble(**

**sc, "Enter price: ");**

**char discount = Validator.getChar(**

**sc, "Enter does the item has discount: ");**

**String company = Validator.getLine(**

**sc, "Enter the company: ");**

**Item item = new Item();**

**item.setCode(code);**

**item.setDescription(description);**

**item.setAmount(amount);**

**item.setPrice(price);**

**item.setDiscount(discount);**

**item.setCompany(company);**

**boolean success = itemDAO.updateItem(item);**

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

**if (success) {**

**System.out.println(description**

**+ " was added to the database.\n");**

**} else {**

**System.out.println("Error! Unable to add item\n");**

**}**

**}**

**}**

**// ItemXMLFile.java**

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

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

**import javax.xml.stream.\*; // StAX API**

**public class ItemXMLFile implements ItemDAO {**

**private String itemsFilename = "items.xml";**

**private File itemsFile = null;**

**public ItemXMLFile() {**

**itemsFile = new File(itemsFilename);**

**}**

**private void checkFile() throws IOException {**

**// if the file doesn't exist, create it**

**if (!itemsFile.exists()) {**

**itemsFile.createNewFile();**

**}**

**}**

**private boolean saveItems(ArrayList<Item> items) {**

**// create the XMLOutputFactory object**

**XMLOutputFactory outputFactory = XMLOutputFactory.newInstance();**

**try {**

**// check the file to make sure it exists**

**this.checkFile();**

**// create XMLStreamWriter object**

**FileWriter fileWriter**

**= new FileWriter(itemsFilename);**

**XMLStreamWriter writer**

**= outputFactory.createXMLStreamWriter(fileWriter);**

**//write the items to the file**

**writer.writeStartDocument("1.0");**

**writer.writeStartElement("Items");**

**for (Item p : items) {**

**writer.writeStartElement("Item");**

**writer.writeAttribute("Code", p.getCode());**

**writer.writeStartElement("Description");**

**writer.writeCharacters(p.getDescription());**

**writer.writeEndElement();**

**writer.writeStartElement("Amount");**

**int amount = (int) p.getAmount();**

**writer.writeCharacters(Integer.toString(amount));**

**writer.writeEndElement();**

**writer.writeStartElement("Price");**

**double price = p.getPrice();**

**writer.writeCharacters(Double.toString(price));**

**writer.writeEndElement();**

**writer.writeStartElement("Discount");**

**char discount = p.getDiscount();**

**writer.writeCharacters(Character.toString(discount));**

**writer.writeEndElement();**

**writer.writeStartElement("Company");**

**writer.writeCharacters(p.getCompany());**

**writer.writeEndElement();**

**writer.writeEndElement();**

**}**

**writer.writeEndElement();**

**writer.flush();**

**writer.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**return false;**

**} catch (XMLStreamException e) {**

**e.printStackTrace();**

**return false;**

**}**

**return true;**

**}**

**public ArrayList<Item> getItems() {**

**ArrayList<Item> items = new ArrayList<Item>();**

**Item p = null;**

**// create the XMLInputFactory object**

**XMLInputFactory inputFactory = XMLInputFactory.newInstance();**

**try {**

**// check the file to make sure it exists**

**this.checkFile();**

**// create a XMLStreamReader object**

**FileReader fileReader**

**= new FileReader(itemsFilename);**

**XMLStreamReader reader**

**= inputFactory.createXMLStreamReader(fileReader);**

**// read the items from the file**

**while (reader.hasNext()) {**

**int eventType = reader.getEventType();**

**switch (eventType) {**

**case XMLStreamConstants.START\_ELEMENT:**

**String elementName = reader.getLocalName();**

**if (elementName.equals("Item")) {**

**p = new Item();**

**String code = reader.getAttributeValue(0);**

**p.setCode(code);**

**}**

**if (elementName.equals("Description")) {**

**String description = reader.getElementText();**

**p.setDescription(description);**

**}**

**if (elementName.equals("Amount")) {**

**String AmountString = reader.getElementText();**

**int amount = Integer.parseInt(AmountString);**

**p.setAmount(amount);**

**}**

**if (elementName.equals("Price")) {**

**String priceString = reader.getElementText();**

**double price = Double.parseDouble(priceString);**

**p.setPrice(price);**

**}**

**if (elementName.equals("Discount")) {**

**String discountString = reader.getElementText();**

**char discount = discountString.charAt(0);**

**p.setDiscount(discount);**

**}**

**if (elementName.equals("Company")) {**

**String company = reader.getElementText();**

**p.setCompany(company);**

**}**

**break;**

**case XMLStreamConstants.END\_ELEMENT:**

**elementName = reader.getLocalName();**

**if (elementName.equals("Item")) {**

**items.add(p);**

**}**

**break;**

**default:**

**break;**

**}**

**reader.next();**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**return null;**

**} catch (XMLStreamException e) {**

**e.printStackTrace();**

**return null;**

**}**

**return items;**

**}**

**public Item getItem(String code) {**

**ArrayList<Item> items = this.getItems();**

**for (Item p : items) {**

**if (p.getCode().equals(code)) {**

**return p;**

**}**

**}**

**return null;**

**}**

**public boolean addItem(Item p) {**

**ArrayList<Item> items = this.getItems();**

**items.add(p);**

**return this.saveItems(items);**

**}**

**public boolean deleteItem(Item p) {**

**ArrayList<Item> items = this.getItems();**

**items.remove(p);**

**return this.saveItems(items);**

**}**

**public boolean updateItem(Item newItem) {**

**ArrayList<Item> items = this.getItems();**

**// get the old item and remove it**

**Item oldItem = this.getItem(newItem.getCode());**

**int i = items.indexOf(oldItem);**

**items.remove(i);**

**// add the updated item**

**items.add(i, newItem);**

**return this.saveItems(items);**

**}**

**}**

**//======================================================================================**

**//StringUtils.java**

**public class StringUtils**

**{**

**public static String padWithSpaces(String s, int length)**

**{**

**if (s.length() < length)**

**{**

**StringBuilder sb = new StringBuilder(s);**

**while(sb.length() < length)**

**{**

**sb.append(" ");**

**}**

**return sb.toString();**

**}**

**else**

**{**

**return s.substring(0, length);**

**}**

**}**

**}**

**//============================================================================**

**//Validator.java**

**import java.util.Scanner;**

**public class Validator {**

**public static String getLine(Scanner sc, String prompt) {**

**System.out.print(prompt);**

**String s = sc.nextLine(); // read the whole line**

**return s;**

**}**

**public static String getString(Scanner sc, String prompt) {**

**System.out.print(prompt);**

**String s = sc.next(); // read the first string on the line**

**sc.nextLine(); // discard the rest of the line**

**return s;**

**}**

**public static int getInt(Scanner sc, String prompt) {**

**boolean isValid = false;**

**int i = 0;**

**while (isValid == false) {**

**System.out.print(prompt);**

**if (sc.hasNextInt()) {**

**i = sc.nextInt();**

**isValid = true;**

**} else {**

**System.out.println("Error! Invalid integer value. Try again.");**

**}**

**sc.nextLine(); // discard any other data entered on the line**

**}**

**return i;**

**}**

**public static int getInt(Scanner sc, String prompt,**

**int min, int max) {**

**int i = 0;**

**boolean isValid = false;**

**while (isValid == false) {**

**i = getInt(sc, prompt);**

**if (i <= min) {**

**System.out.println(**

**"Error! Number must be greater than " + min);**

**} else if (i >= max) {**

**System.out.println(**

**"Error! Number must be less than " + max);**

**} else {**

**isValid = true;**

**}**

**}**

**return i;**

**}**

**public static double getDouble(Scanner sc, String prompt) {**

**boolean isValid = false;**

**double d = 0;**

**while (isValid == false) {**

**System.out.print(prompt);**

**if (sc.hasNextDouble()) {**

**d = sc.nextDouble();**

**isValid = true;**

**} else {**

**System.out.println("Error! Invalid decimal value. Try again.");**

**}**

**sc.nextLine(); // discard any other data entered on the line**

**}**

**return d;**

**}**

**public static double getDouble(Scanner sc, String prompt,**

**double min, double max) {**

**double d = 0;**

**boolean isValid = false;**

**while (isValid == false) {**

**d = getDouble(sc, prompt);**

**if (d <= min) {**

**System.out.println(**

**"Error! Number must be greater than " + min);**

**} else if (d >= max) {**

**System.out.println(**

**"Error! Number must be less than " + max);**

**} else {**

**isValid = true;**

**}**

**}**

**return d;**

**}**

**public static char getChar(Scanner sc, String prompt) {**

**char c = ' ';**

**boolean isValid = false;**

**while (isValid == false) {**

**System.out.print(prompt);**

**if (sc.hasNext()) {**

**c = sc.findInLine(".").charAt(0);**

**if (c == 'y' || c == 'n') {**

**isValid = true;**

**} else {**

**System.out.println("Error! Invalid decimal value. Try again.");**

**}**

**}**

**sc.nextLine(); // discard any other data entered on the line**

**}**

**return c;**

**}**

**}**

**//===================================================================================**

**//ItemConstants.java**

**public interface ItemConstants**

**{**

**int CODE\_SIZE = 4;**

**int DESCRIPTION\_SIZE = 20;**

**}**

**//====================================================================================**

**C’est Finis**