CSCI-3901-Assignment5

Name:	Pratheep Kumar Manoharan	
Banner Id:	B00837436	
Files:	*constants:*	ConnectionProperties.java, ConstantsClass.java
	control:	Controller.java, ControllerClass.java
	db:	Queries.java, DBHandler.java, DBHandlerClass.java
	plainobjects:	Address.java, Category.java, Customer.java, Product.java, Supplier.java
	xml:	XMLGenerator.java, XMLGeneratorClass.java

MainClass.java

CSCI-3901-Assignment 5.pdf
CSCI-3901-Assignment 5-Argument.pdf

Objective:

To learn and work on how to access SQL through Java. To gain some exposure to XML by producing the output in XML format.

Assumptions:

1. Output file name should be of full path and ends with XML.
2. The dates provided by user should not be a future date.
3. Start date should be less than end date.
4. Data from database will not be changed in case of any special characters present for instance the carriage return or extra spaces in database will be reflected as it is in xml file as well.
5. Cost field is rounded off to two decimals.
6. Cost field is calculated as shown in lab 7. The sum of product of quantity and unitprice in orderdetails table.

Dependencies used:

1. mysql-connector-java-8.0.19.jar – to establish connection between Java and SQL.

Program Design:

1. First, the program validates the inputs.
2. If even one of the inputs is invalid, then the program prints appropriate message and ends there.
3. If all inputs are valid, then data are fetched from database for the date range.
4. The data are converted into XML file in requested format.

Queries:

1. Customer information:
   a. Customer name and address details are taken from customers table.
   b. Number of orders value is taken from the unique count of order ids from the orders table joined with customers table by customer id column and grouped by customer id in the mentioned date range.
   c. Order value is calculated from the sum of product of quantity and unit price of the products from the orderdetails table joined with orders table by order id grouped by customer ids in the mentioned date range.

2. Product information:
   a. Category name is taken from categories table.
   b. Product name is taken from products table joined with customers table by product id column.
   c. Supplier name is taken from suppliers table joined with products table by supplier id column.
   d. Units sold is calculated from the sum of quantities taken from the orderdetails table joined with orders table connected with products table by product id in the date range of order date.
   e. Sale value is calculated from the sum of product of quantities and unit price taken from the orderdetails table.

3. Supplier information:
   a. Supplier name and address details are taken from suppliers table.
   b. Supplier id is used to join suppliers and products tables.
   c. Number of products is calculated from the sum of quantities from the orderdetails table joined with orders table by order id and grouped by supplier id in the mentioned date range.
   d. Product value is calculated from the sum of product of quantity and unit price of the products from the orderdetails table.

Java Object classes:

Address.java:

Address object is created with all the attributes of the address.

Customer.java:

1. Customer information are stored in the Customer object.
2. Address object is added as a field in the Customer object.

Supplier.java:

1. Supplier information are stored in the Supplier object.
2. Address object is added as a field in the Supplier object.

Product.java:

Product object is created with all the attributes of the product.

Category.java:

Category object contains category name and the list of the products in that category.

ConnectionProperties.java:

Class is used to store the database credentials which is used in DBHandlerClass to connect to the database.

ConstantsClass.java:

Class is used to store all the constants of the program.

Technical Code Flow:

MainClass.java:

1. The main method is the starting point of the program.
2. The program gets the date range and the output filename in this class.
3. The flow of the program is controlled by this class.
4. Pass the inputs to the validateInput method to validate.
5. If any one of the inputs is invalid, program prints an appropriate message and ends.
6. If all the inputs are valid, then the data are fetched from database by calling getData method.
7. The data are converted into XML format by calling formXMLString method.
8. The data are written into the output file by calling writeToOutputFile method.
9. The output file will be validated for valid xml format by calling validateXML method.

Controller.java:

Interface to provide bridge between the main method of the program and the rest of the program.

ControllerClass.Java:

1. ControllerClass is used as a connector as well as separator between the MainClass and the rest of the program.

2. printString method gets a string as an input and prints the input to the user.

3. validate method:

    a. validate method gets the start date, end date and output file name as inputs.
    b. Basic string validations are done on the inputs at first.

c. Valid strings are trimmed to remove the unwanted spaces.
d. Output file name is validated whether it is of xml file type.
e. Dates are checked whether they are of valid format.
f. Dates are compared with current date as the future dates are invalid as per assumption.
g. End date should be greater than start date.
h. Based on the validation status, the boolean value is returned to the main method.
i. ParseException is used to catch the date parsing exception while validation the format.
j. Exception is used to catch all the rest of the exceptions that may occur.

4. The getData method is used to fetch the information from database in the date range.

5. The formXMLString method is used to form the XML content as string.

6. stringValidator method is used to validate whether the inputted string is null or empty.

7. dateConverter method is used to validate whether the string input is of valid date format. If the string is valid then the date is returned.

8. writeToOutputFile method
   a. writeToOutputFile method is used to write the final xml content into the output file.
   b. Gets the filename with path and xml content as inputs.
   c. Creates the output stream and write the content into the file.
   d. Catches in case of any exceptions.
   e. Closes the output stream in finally block.
   f. Returns boolean value.

9. validateXML method
   g. validateXML method is used to validate the final xml content.
   h. Gets the xml content as input.
   i. Creates the document builder and validates the content.
   j. Catches in case of any exceptions.
   k. Returns boolean value.

Queries.java:

1. The class is used to build queries.

2. The static parts of the queries are stored in the form of string values.

3. buildCustomerQuery, buildProductQuery and buildSupplierQuery methods get the date range as inputs and form the complete query by combining the static parts and the dynamic parts. The final query is returned from the method to the DBHandlerClass to fetch the customer, product and supplier information respectively from the database.

DBHandler.java:

Interface between the Controller and the database layer of the program.

DBHandlerClass.java:

1. Database activities of the program are done in this class.

2. getFieldNameFromMethod gets the method name as input and returns the field name as output.
3. getData method:

    a. Database credentials are fetched from ConnectionProperties class.
    b. Creates the database connection using the dependency jar.
    c. Establishes the connection in the same way as lab.
    d. Query is formed and retrieved from the Queries class.
    e. Data is retrieved with all the required attributes customer, product and supplier within the date range.
    f. Separate try blocks are used for each query so that the error in one does not affect others.
    g. Data is converted into the java object for easy processing.
    h. Single map is formed with all the details.

4. convertDataIntoObj method:

    a. Gets the resultSet from database and the object name of the resultSet as inputs.
    b. Using reflection method, all the methods in the object are stored in an array.
    c. Based on the object name value, object is created.
    d. All the setter methods in the class are looped to set the value into the object.
    e. The method name, field name, value, value type, and object are passed to the reflection method to set the value into the object.
    f. Since the address field and product list fields are itself an object and list of objects, they are handled in the separate if blocks in the code.
    g. Address object is created, and the values of the address object are set in the similar way of the main object.
    h. Once all the values are set into the address object, the address object set into the main object as a field.
    i. Like the address object, product object is formed and once all the products of the same category are added into a list, the list is set into the category object.

5. getSetterMethodForField method:

    a. Gets the object, fieldname, and Class type as inputs.
    b. Using reflection method, the method details are returned to the called method.

XMLGenerator.java:

Interface between the Controller and the xml generator class.

XMLGeneratorClass.java:

1. prepareXML method:
    a. Gets the customer object list, category object list, supplier object list and date range as inputs.
    b. StringWriter object is used to stream the xml content as a string buffer.
    c. XMLOutputFactory object is used to define XMLStreamWriter.
    d. XMLStreamWriter is used to write to XML.
    e. As per specified structure, the xml content is written into the string buffer.
    f. Basic elements of the xml are designed in this method.
    g. The object information is passed to formXMLFromObject method.

2. formXMLFromObject method

    a. Gets the same XMLStreamWriter object and the list of objects to write into the XML file as inputs.
    b. Since the order of the XML tags are predefined, the fields are stored into an array and it is written into the XML file in the same predefined order.
    c. The tag name is manipulated from the method name.
    d. The normal fields of the main object are written using the reflection method.
    e. The tag name and the field value are written into the string buffer.
    f. The tags inside the address tag are retrieved from the address object which in turn is a field in customer and supplier objects.
    g. The tags inside the product tag are retrieved from the product object which in turn is stored as a list of products of same category in the category object.

3. getMethod method

    a. Gets the object, methodName and class array as inputs.
    b. Using reflection method, returns all the details of the method.

4. getPropertyNameFromMethod method

    a. Gets the methodName as input.
    b. Forms the xml tag name from the methodName.
    c. Two-word method names are separated by "_" (underscore) in the requirement.
    d. With this method, we can change manipulate the tag names automatically.

Maintainability of the program (Design elements for minimal future change):

1. In case if we want to add new tag, we just need to make changes in three places.
2. We need to add the tag name as a field in the corresponding object say new field under product tag then a new field in product object and its setter and getter methods.
3. We need to add the column in the corresponding query with the field name as the output column header so that the column value will be set into the object using reflection method.
4. We need to add the getter method name of the new field in the ConstantClass array field in the expected order so that the tag is formed in the correct sequence.
5. Same steps to remove a field from the XML as well.
6. All other details of the program are designed in the generalised way to have less maintainability effort.

Test cases:

S.No	**Test Scenario**	**Expected result**	**Actual result**
Input validations			
1	Give the start date format as dd-mm-yyyy	validate method should return false	as expected
2	Give the start date format as mm-dd-yyyy	validate method should return false	as expected
3	Give the start date format as dd-MMM-yyyy	validate method should return false	as expected
4	Give the start date format as MMM-dd-yyyy	validate method should return false	as expected
5	Give the end date format as dd-mm-yyyy	validate method should return false	as expected
6	Give the end date format as mm-dd-yyyy	validate method should return false	as expected
7	Give the end date format as dd-MMM-yyyy	validate method should return false	as expected
8	Give the end date format as MMM-dd-yyyy	validate method should return false	as expected
9	Give the end date format as yyyy-mm-dd	validate method should return true	as expected
10	Give null as ouput file name	validate method should return false	as expected
11	Give wrong path in output file name	validate method should return false	as expected
12	Give non xml file name	validate method should return true	as expected
13	Give end date less than start date	validate method should return false	as expected
14	Give end date equal to start date	validate method should return true	as expected
15	Give start date as a future date	validate method should return false	as expected
16	Give end date as a future date	validate method should return false	as expected
boundary cases			
17	try to get data without vpn connection to simulate no db connection scenario	program should handle the exception	as expected

database cases			
18	null values in between	empty tags should be in place	as expected
19	error in customer query	customer information will not be fetched but other information will be produced	as expected
20	error in product query	product information will not be fetched but other information will be produced	as expected
21	error in supplier query	supplier information will not be fetched but other information will be produced	as expected
22	error in all queries	no output empty	as expected
23	error in customer and product queries	only supplier info	as expected
24	error in product and supplier queries	only customer info	as expected
25	error in customer and supplier queries	only product info	as expected
xml test cases			
26	xml structure when no records in the time range	only basic tags	as expected
27	xml structure when records are there	perfect structure with spacing and carriage returns	as expected
28	xml structure when only one customer record	xml structure should be proper	as expected
29	xml structure when more than one customer records	xml structure should be proper	as expected
30	xml structure when only one category record with only one product record	xml structure should be proper	as expected
31	xml structure when only one category record with more than one product records	xml structure should be proper	as expected
32	xml structure when more than one category records with only one product record in each category	xml structure should be proper	as expected
33	xml structure when more than one category records with more than one product records in each category	xml structure should be proper	as expected
34	xml structure when only one supplier record	xml structure should be proper	as expected
35	xml structure when more than one supplier records	xml structure should be proper	as expected

36	xml structure when null values in between	xml structure should be proper	as expected
additional cases			
37	add product id under product tag	xml structure should be proper	as expected
38	add supplier id under supplier tag	xml structure should be proper	as expected
39	remove these tags	xml structure should be proper	as expected
40	validate the output file with incorrect output	basic xml validation	as expected
41	validate the output file with correct output	values of customer, product and supplier should tally	as expected