Criterion C

Development

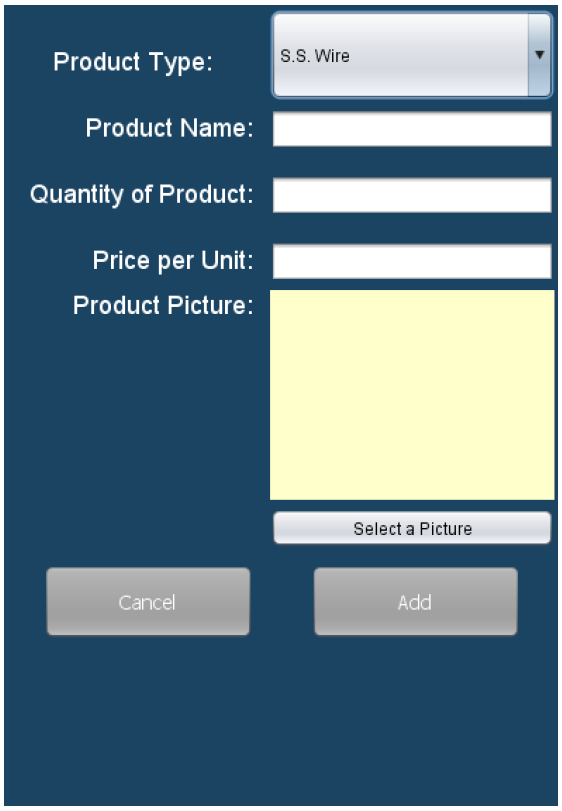
Table of Contents

SQL Queries

Adding Data to an instance

This Query is used to insert records into the table in a database. This is used in several places like add users, add product, add category, add customer. The below image shows add product code where there are fields like ID, Name, Description, Image etc. Error handling is also used through try and catch. I have made classes separately and the used call by referencing in the add product form. I have similar format for other forms as well

```
INSERT INTO `category` (`id`, `name`) VALUES ('2', 'SS Wire')
```



Deleting Data from an instance

This query is used to delete records from a table in a database. This is used in several places as well like delete product, delete customer, delete user. The below screenshot is of delete query in customer form. It follows a similar format as insert query where error handling is used, and classes are made separately and then call by referencing is used. This is also used for other delete queries.

```
DELETE FROM `customer` WHERE `customer`.`id` = 1;
```

Editing Data to an instance

This query is used for updating and changing pre-existing records. This is used in various forms such as Update Product, Edit customers, Edit category etc. The below screenshot is of the Edit Category button. It uses error handling through try and catch and classes are made which are called by referencing.

```
UPDATE `order_tbl` SET `customer_id` = '2' WHERE `order_tbl`.`id` = 1
```

Searching

This query is used to find a specific record in a file or database. I have used searching in the products form only for searching for particular products. I have used a user defined function populateJtable and then used call by values for the search function. Rows and colNames are the formal parameters and mmd is the object name.

```
SELECT * FROM `product` WHERE price = 155;
```

Polymorphism

I have used Method overriding which allows a sub-class to inherit a method from a super-class, when a method in a subclass uses same name, parameters and return type as the super class, then the method of sub-class overrides the super-class. Overriding is a part of run-time polymorphism. The below screenshot shows the sub-class part which overrides the super-class. These are used to get column count, row count and get value at with formal parameters as rowIndex and columnIndex.

```java
    @Override
    public int getRowCount() {

        return this.rows.length;

    }

    @Override
    public int getColumnCount() {

        return this.columns.length;

    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {

        return this.rows[rowIndex][columnIndex];

    }
```

Inheritance

In object-oriented programming, inheritance is used for deriving methods from existing class to the other. The keyword 'extends' is used for inheritance. I have used inheritance in my program mainly to reuse old code and also overriding. I have used hierarchical inheritance where one super class is inherited by many sub classes which makes it a complex structure, along with this single inheritance and multilevel inheritance is also used.

```java
public class Add_Product_Form extends javax.swing.JFrame {
```

```java
public class All_Orders_Form extends javax.swing.JFrame {
```

```java
public class Edit_Product_Form extends javax.swing.JFrame {
```

Encapsulation

It is one of the basic concepts of object-oriented programming, it is used to wrap the data and code into a single unit. The variables of a particular class are hidden from other classes, also called Data Hiding. This helps to modify the fields of the class to be read-only or write-only. I have also used both methods under encapsulation getters and setters, which are used to access and update private classes. Encapsulation has also allowed me to change certain parts of classes without affecting the others.

```java
public Customer(){}


public Customer(Integer ID, String FNAME, String LNAME, String TEL, String EMAIL)
{
    this.id = ID;
    this.first_name = FNAME;
    this.last_name = LNAME;
    this.tel = TEL;
    this.email = EMAIL;
}

public Integer getId() {
    return id;
}
public void setId(Integer id) {
    this.id = id;
}
public String getFirst_name() {
    return first_name;
}

public void setFirst_name(String first_name) {
    this.first_name = first_name;
}

public String getLast_name() {
    return last_name;
}

public void setLast_name(String last_name) {
    this.last_name = last_name;
}

public String getTel() {
    return tel;
}

public void setTel(String tel) {
    this.tel = tel;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
```

Exception Handling

This is used to show errors in inputs by the user, in java errors are known as exceptions. These generate an error message whenever an invalid input is given by the user. The keywords used for this are 'try' and 'catch'. The try statement tests for errors and the catch defines the process after try. The below screenshot of code shows exception handling for the delete customers button.

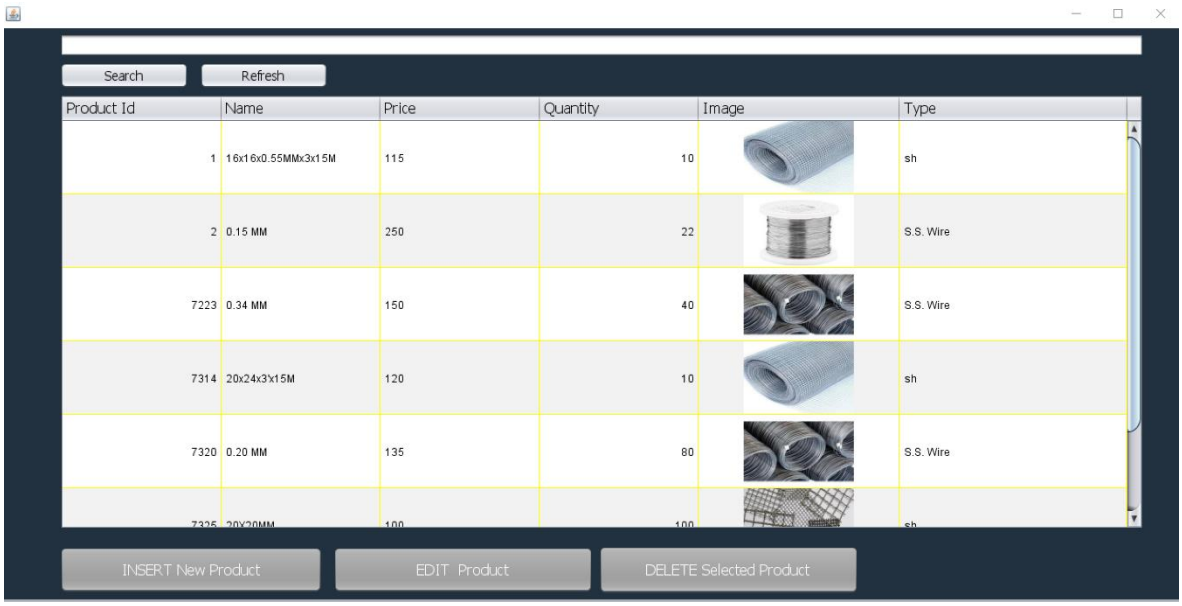
```java
private void jButton_DELETE_ActionPerformed(java.awt.event.ActionEvent evt) {

    try{
        Integer id = Integer.valueOf(jTextField_ID.getText());
        CLASS.Customer.deleteCustomer(id);
        populateJtable();
    }catch(Exception ex){
            JOptionPane.showMessageDialog(null, "Select a Customer Before Removing", "No Customer Selected", 1);
            }

}
```

It will show an error if no customer is selected while deleting data values.

Two-dimensional Array

I have used 2D arrays in various places. The below picture shows a 2D array of rows and columns where the array for rows is two-dimensional and I have also used a loop for the first parameter for the 2D array of rows. 2D arrays are similar to normal arrays but are organised structure as matrix with rows and columns.

```java
public void populateJtable(String val){

    CLASS.Product prd = new CLASS.Product();
    ArrayList<CLASS.Product> ProductList = prd.productsList(val);

    String[] colNames = {"Id","Name","Price","Quantity","Image","Description","Category"};
    Object[][] rows = new Object[ProductList.size()][7];

    for(int i = 0; i < ProductList.size(); i++){
        rows[i][0] = ProductList.get(i).getId();
        rows[i][1] = ProductList.get(i).getName();
        rows[i][2] = ProductList.get(i).getPrice();
        rows[i][3] = ProductList.get(i).getQuantity();

                    ImageIcon pic = new ImageIcon(new ImageIcon
                                    (ProductList.get(i).getPicture())
                                     .getImage()
                                     .getScaledInstance(120, 80, Image.SCALE_SMOOTH));

        rows[i][4] = pic;

        rows[i][5] = ProductList.get(i).getDescription();
        rows[i][6] = ProductList.get(i).getCategoryName(ProductList.get(i).getId());

    }

    CLASS.MyTableModel mmd = new CLASS.MyTableModel(rows, colNames);
    jTable_Products.setModel(mmd);
    jTable_Products.setRowHeight(80);
    jTable_Products.getColumnModel().getColumn(5).setPreferredWidth(150);
    jTable_Products.getColumnModel().getColumn(4).setPreferredWidth(120);
}
```



Additional Libraries

I have used a lot of additional libraries and imported a lot of packages for my program and code to function according to my success criteria. Some of these include MySQL connector for java, jdatechooser which helped me include a calendar in my program and jcalendar. Below is an example of some other libraries imported.

```
import java.awt.Dimension;
import java.awt.Font;
import java.awt.font.TextAttribute;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
```

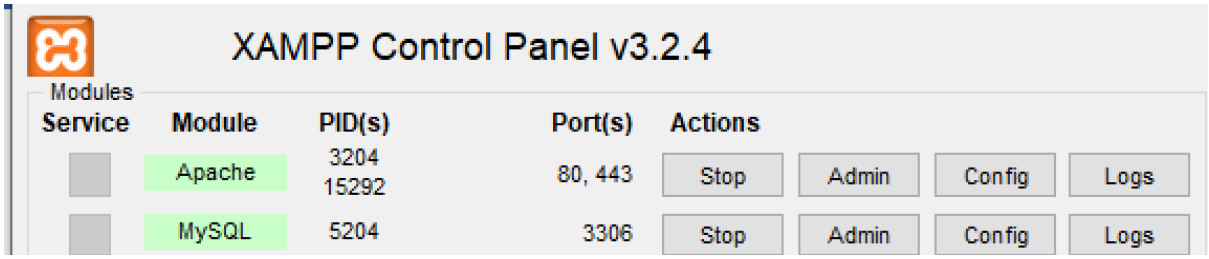
GUI

The GUI was used through the generated code from the NetBeans software which is generated automatically according to the inputs in the design window. For example, in the login window, Arial font was used with 18 and 36 as the font sizes. Different colour schemes

were used according to the company logo where the background and foreground text varied in colour.

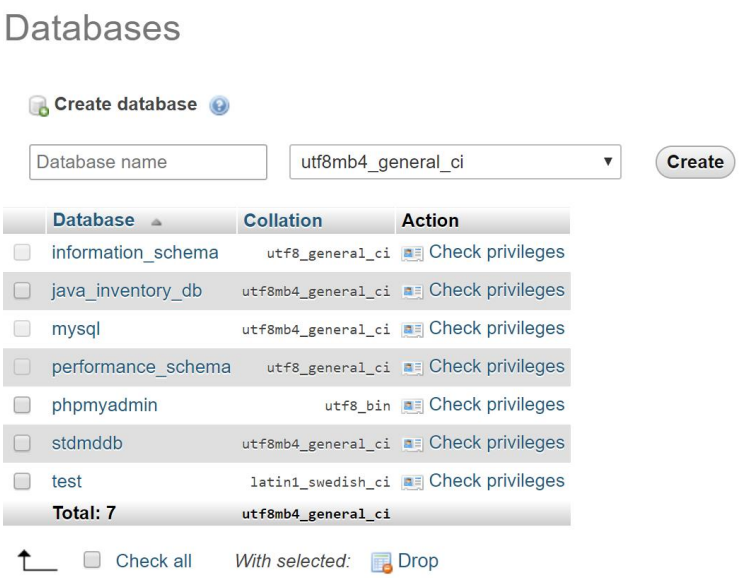
Creating the database

1. Log on to the phpMyAdmin server using xampp control panel



2. Use the create new database option to create a database

3. A new window opens where the name of the database needs to be added according to the name in the Code.



4. It creates a new database where tables need to add either using the add tables option or using SQL tab where the queries to create a new table can be entered.

5. Once the queries are done press go, the system will debug and create a table

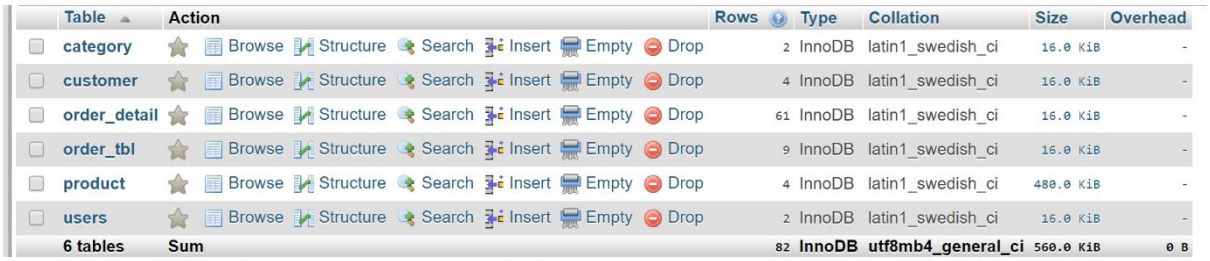
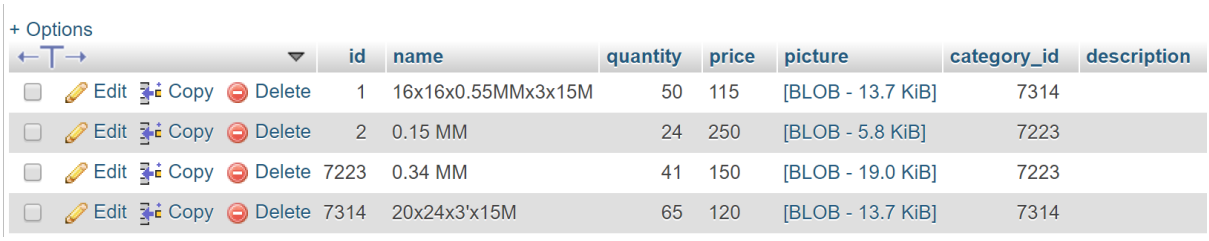


Table	Action	Rows	Type	Collation	Size	Overhead
category	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16.0 KiB	-
customer	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	16.0 KiB	-
order_detail	Browse Structure Search Insert Empty Drop	61	InnoDB	latin1_swedish_ci	16.0 KiB	-
order_tbl	Browse Structure Search Insert Empty Drop	9	InnoDB	latin1_swedish_ci	16.0 KiB	-
product	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	480.0 KiB	-
users	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16.0 KiB	-
6 tables	Sum	82	InnoDB	utf8mb4_general_ci	560.0 KiB	0 B

6. All tables can be modified using fields, type and length



+ Options

	id	name	quantity	price	picture	category_id	description
Edit Copy Delete	1	16x16x0.55MMx3x15M	50	115	[BLOB - 13.7 KiB]	7314	
Edit Copy Delete	2	0.15 MM	24	250	[BLOB - 5.8 KiB]	7223	
Edit Copy Delete	7223	0.34 MM	41	150	[BLOB - 19.0 KiB]	7223	
Edit Copy Delete	7314	20x24x3'x15M	65	120	[BLOB - 13.7 KiB]	7314	

Connecting the database

The following JAVA code was used to connect the MySQL database from phpMyAdmin to the program:

```java
package CLASS;

import java.sql.Connection;
import java.sql.DriverManager;


public class DB_INFO {

    private static String dbname = "java_inventory_db";
    private static String username = "root";
    private static String password = "";

       static Connection con=null;
    public static Connection getConnection()
    {
        if (con != null) return con;
        // get db, user, pass from settings file
        return getConnection(dbname, username, password);
    }

    private static Connection getConnection(String db_name,String user_name,String password)
    {
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            con=DriverManager.getConnection("jdbc:mysql://localhost/"+db_name+"?user="+user_name+"&password="+password);
            System.out.println("connected");
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        return con;
    }
```