

## **BITP 3113: OBJECT ORIENTED PROGRAMMING**

### **Mini Project Deliverable**

**Project Title:** Art Gallery System

**Team members:**

<b>Name</b>	<b>Matric</b>	<b>Sek/ Kump</b>	<b>Program (BITC/I/Z)</b>
Muhd Farhan Bin Mohd Noor	B032010270	2	BITI
Abdullah Harith bin Jamadi	B032010181	2	BITI
Amirul ian rashidee bin mohd jasni	B032010041	2	BITI
Muhammad Muhazill Daniel Bin Hassan	B032010290	2	BITI
Rusydi Nasution Bin Riduan	B032010215	2	BITI

Please paste your diagram/code snapshot after each item number

**Diagram:**

- 1) **UML diagrams**
- 2) **GUI**

**Code snapshot:**

- 1) **Inheritance**
- 2) **Polymorphism**
- 3) **Interface**
- 4) **Abstract**
- 5) **Java Collection: List / Set / Map**
- 6) **Exception Handling (Create your own)**
- 7) **Connecting to database**

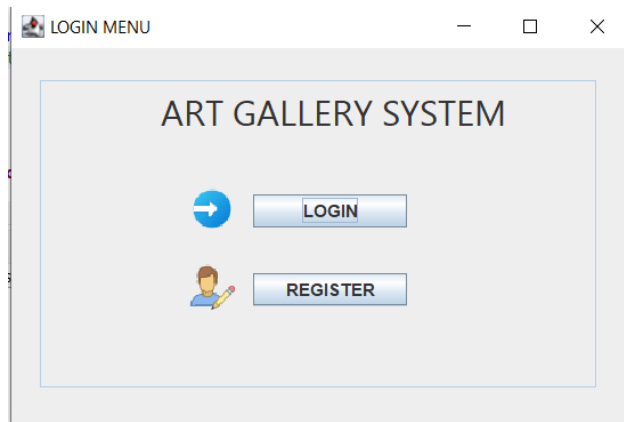
```

classDiagram
    package database.com {
        class Gateway
    }
    package database.art {
        class Artist
        class Album
        class Track
    }
    package database.album {
        class Album
        class Track
    }
    package database.exception {
        class Exception
        class SQLException
    }
    package database.util {
        class StringUtil
        class DateUtil
    }
    package database.view {
        class ArtistView
        class AlbumView
        class TrackView
    }
    package database.service {
        class ArtistService
        class AlbumService
        class TrackService
    }
    package database.controller {
        class ArtistController
        class AlbumController
        class TrackController
    }
    package database.model {
        class ArtistModel
        class AlbumModel
        class TrackModel
    }

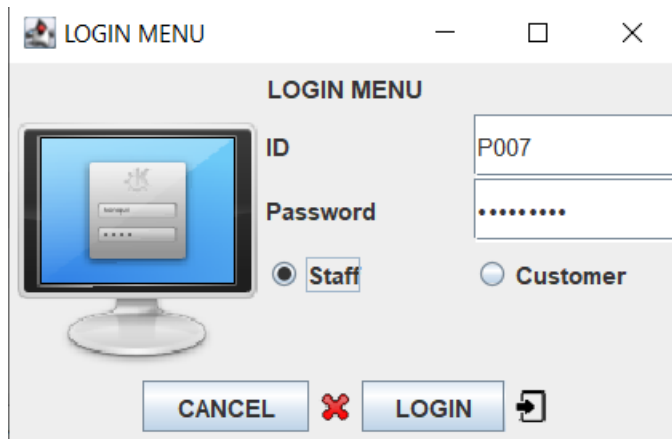
    database.com.Gateway <|-- database.art.Artist
    database.com.Gateway <|-- database.album.Album
    database.com.Gateway <|-- database.view.TrackView
    database.art.Artist <|-- database.art.ArtistImpl
    database.art.Album <|-- database.album.AlbumImpl
    database.album.Album <|-- database.album.AlbumImpl
    database.album.Track <|-- database.album.TrackImpl
    database.exception.Exception <|-- database.exception.SQLException
    database.util.StringUtil <|-- database.util.StringUtilImpl
    database.util.DateUtil <|-- database.util.DateUtilImpl
    database.view.ArtistView <|-- database.view.ArtistViewImpl
    database.view.AlbumView <|-- database.view.AlbumViewImpl
    database.view.TrackView <|-- database.view.TrackViewImpl
    database.service.ArtistService <|-- database.service.ArtistServiceImpl
    database.service.AlbumService <|-- database.service.AlbumServiceImpl
    database.service.TrackService <|-- database.service.TrackServiceImpl
    database.controller.ArtistController <|-- database.controller.ArtistControllerImpl
    database.controller.AlbumController <|-- database.controller.AlbumControllerImpl
    database.controller.TrackController <|-- database.controller.TrackControllerImpl
    database.model.ArtistModel <|-- database.model.ArtistModelImpl
    database.model.AlbumModel <|-- database.model.AlbumModelImpl
    database.model.TrackModel <|-- database.model.TrackModelImpl
  
```

# GUI:

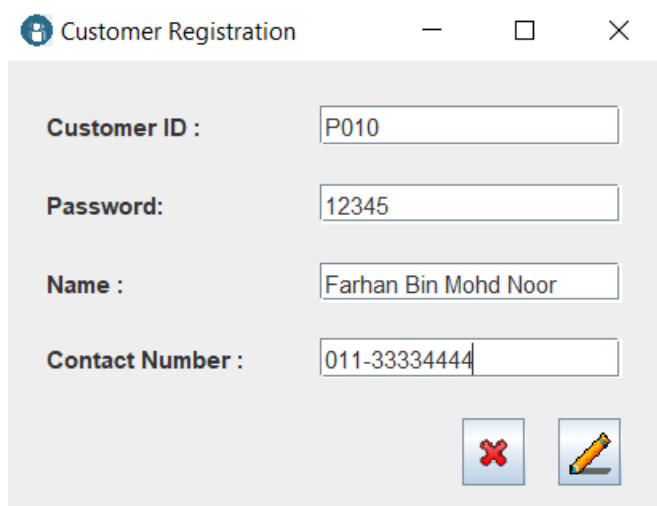
## Main Menu:



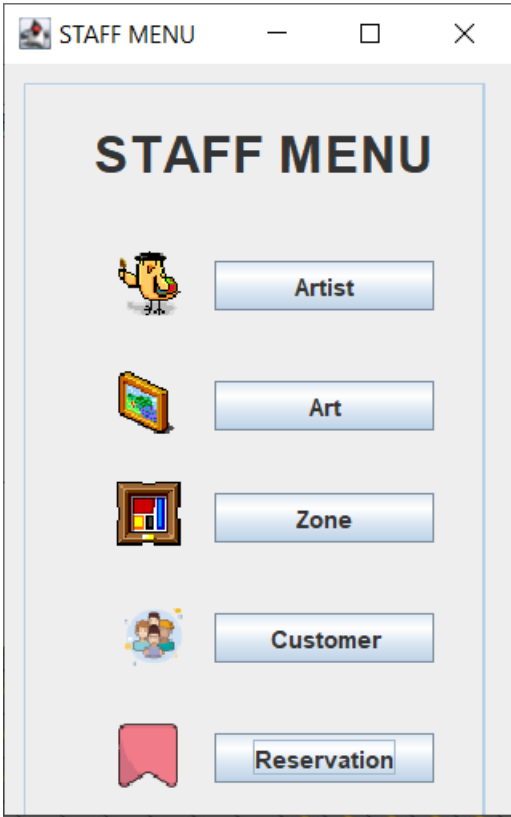
## Login Menu:



## Customer Register Menu:



**After Login As A Staff In Staff Menu:**



**Artist Menu**

**Artist**

Register

Update

Search

Artist ID	Name	Country	Contact Number
1001	Linda Cantrell	Thailand	03 8739-1377
1002	Shayla Rocha	Russia	033343-7317
1003	Aairah John	Cameroon	05 5485922
1004	Ahsan Simmonds	Oman	089762310
1005	Oisin Blake	Suriname	039101-2289
1006	Maya Crouch	Thailand	06677-5586
1007	Nazia Salter	Abkhazia	03-5633-3073
1008	Hakeem Li	Mexico	0321415077
1009	Kiefer Clark	Egypt	03 21445481
1010	Lily-Mae Woodcock	Saint Barthelemy	011-8886 7881
1011	Aaken Li	Hungary	0378800854

**Art Menu**

**Art**

Register

Update

Search

Art ID	Name	Description	Publish Date	Price	Artist ID	Zone ID
1001	Moana Linda	Elegant	1950-12-10	1000000.0	1001	1001
1002	Venus Wra...	Luxurious	1970-03-06	500450.0	1002	1002
1003	John The ...	Elegant	2001-06-01	1000050.0	1003	1005
1004	Gotham As...	Elegant	2005-12-12	500000.0	1004	1006
1005	Abstract S...	Style	2010-03-12	350000.0	1005	1003
1006	Hamlet	Luxurious	1977-10-03	2300250.0	1006	1005
1007	Terracotta ...	Luxurious	1980-11-02	5000300.0	1006	1004
1008	Queen Eliz...	Style	1997-04-03	750000.0	1001	1007
1009	Iron Will Fo...	Elegant	2005-12-07	860560.0	1009	1005
1010	Byzantine ...	Elegant	2001-06-05	450400.0	1011	1005
1011	The Siege ...	Luxurious	1967-12-03	3000750.0	1011	1004
1012	Queens of	Royalty	1969-04-20	4500000.0	1001	1005

**Zone Menu**

**Zone**

Search

Zone ID	Zone Name
1001	Renaissance
1002	Sculpture
1003	Modern Art
1004	Literature
1005	Classical
1006	Gothic
1007	Victorian

**Customer Menu**

**Customer**

Update

Search

Customer ID	Name	Password	Contact Number
P001	Addison Jacobson	123456789	011-4158 4223
P002	Jules Vo	password	011-7557 0393
P004	Yvie Griffiths	123asz	011-1677 6279
P005	Madelaine Pike	qwer1234	018-472 6448
P006	Yvie Griffiths	987654321	011-8889 6118

**Reservation Menu**

**ART RESERVATION**

ART ID 1001




Search


CONFIRM RESERVATION

CANCEL RESERVATION

Person ID	Art ID	Date	Status
P001	1001	2020-12-12	SOLD

## Artist Sub-Menu:


Artist Registration		Artist Update	
Artist ID :	<input type="text" value="1012"/>	Insert Artist ID :	<input type="text" value="1001"/> 
Name :	<input type="text" value="Aaron Lee"/>	Name :	<input type="text" value="Linda Cantrell"/>
Country :	<input type="text" value="USA"/>	Country :	<input type="text" value="Thailand"/>
Contact Number :	<input type="text" value="023-24445555"/>	Contact Number :	<input type="text" value="03 8739-1377"/>
			

Artist Search	
Insert Artist ID :	<input type="text" value="1001"/> 
Name :	<input type="text" value="Linda Cantrell"/>
Country :	<input type="text" value="Thailand"/>
Contact Number :	<input type="text" value="03 8739-1377"/>


## Art Sub-Menu:


Art Registration	
Art ID :	<input type="text" value="1031"/>
Name :	<input type="text" value="WinterWunderland"/>
Price :	<input type="text" value="400000"/>
Publish Date :	<input type="text" value="2001-10-13"/>
Artist ID :	<input type="text" value="1003"/>
Zone ID :	<input type="text" value="1003"/>
Description :	<input type="text" value="Elegant"/>


Art Search	
Insert Art ID :	<input type="text" value="1030"/> 
Name :	<input type="text" value="Visitor"/>
Price :	<input type="text" value="660000.0"/>
Publish Date :	<input type="text" value="2003-04-04"/>
Artist ID :	<input type="text" value="1004"/>
Zone ID :	<input type="text" value="1002"/>
Description :	<input type="text" value="Style"/>

Art Update	
Insert Art ID :	<input type="text" value="1030"/> 
Name :	<input type="text" value="Visitor"/>
Price :	<input type="text" value="660000.0"/>
Publish Date :	<input type="text" value="2003-04-04"/>
Artist ID :	<input type="text" value="1004"/>
Zone ID :	<input type="text" value="1002"/>
Description :	<input type="text" value="Style"/>



## Zone Sub-Menu:

Zone Search	
Insert Zone ID :	<input type="text" value="1007"/> 
Name :	<input type="text" value="Victorian"/>

## Customer Sub-Menu:

Customer Update

Insert Customer ID :

Name :

Password :

Contact Number :

Customer Search

Insert Customer ID :

Name :

Password :

Contact Number :

## If Login as A Customer In Login Menu:

CUSTOMER MENU

GALLERY

Search

Search By

☐ Art

☐ Artist

Sort By Price

Ascending

Descending

RESERVATION

View

ART ID :

Cancel

Reserve

ART

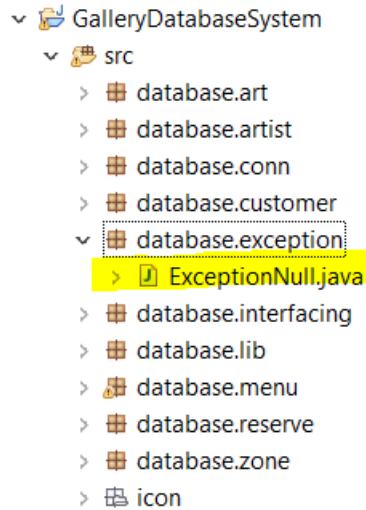
AllRenaissanceSculptureModern ArtLiteratureClassicalGothicVictorian

ART ID	NAME	DESCRIPTION	PUBLISH DATE	PRICE (\$)	ARTIST ID	ZONE ID
1007	Terracotta Squad	Luxurious	1980-11-02	5000300.0	1006	1004
1008	Queen Elizabeth Victoria	Style	1997-04-03	750000.0	1001	1007
1009	Iron Will Forged	Elegant	2005-12-07	860560.0	1009	1005
1012	Queens of Panic	Royalty	1969-04-20	4500000.0	1001	1005
1013	Abstract Message	Colourful	2000-06-05	500000.0	1004	1003
1014	la Couverture Divin	Style	2003-04-13	7000000.0	1006	1004
1015	Infini	Wonderful	2012-03-14	100000.0	1007	1005
1016	Garden of Freedom	Elegant	2010-03-12	750000.0	1004	1005
1017	Veiled Beauty	Luxurious	1951-05-22	300000.0	1001	1001
1018	Refusal	Dark	1999-04-03	1050000.0	1005	1006
1019	Oath	Divine	1977-04-03	1350000.0	1003	1004
1020	Euphoric Switch	Style	1987-04-03	3050000.0	1004	1003
1021	Gracious Battle	Style	1997-04-03	3510000.0	1011	1004
1022	Death of Ice	Divine	1989-04-03	3750000.0	1001	1001
1023	Awoken Fall	Dark	2006-04-03	4350000.0	1010	1002
1024	Brain	Dark	2004-04-03	500000.0	1007	1005
1025	Euphoric Sister	Dark	1967-04-16	787000.0	1004	1004
1026	Mask	Dark	1996-04-26	900000.0	1006	1005
1027	Wounds	Dark	1977-07-07	270000.0	1008	1004
1028	Winter	Elegant	2001-11-15	1000000.0	1006	1003
1029	Fall of Evil	Fear	2002-01-03	980000.0	1005	1002
1030	Visitor	Style	2003-04-04	660000.0	1004	1002



# 1) Inheritance

Located Here:



Code Snippet:

```
1 package database.exception;
2
3 import javax.swing.JOptionPane;
4
5 public class ExceptionNull extends Exception{
6
7     //Deserialization uses this number to ensure that a
8     //loaded class corresponds exactly to a serialized object
9     static final long serialVersionUID = 1L;
10
11     public ExceptionNull(String message){
12         super(message);
13         JOptionPane.showMessageDialog(null, message);
14     }
15
16 }
```

## 2) Polymorphism

Located Here:

```
▼ database.lib
  > Art.java
  > Artist.java
  > Person.java
  > Reserve.java
  > Zone.java
```

Coding Snippet Template(Person.java):

```
14 public class Person implements PersonInterface{
15
16     private String PersonID;
17     private String Password;
18     private String Name;
19     private String Contact;
20     private String Role;
21
22     public String getPersonID() {
23         return PersonID;
24     }
25     public void setPersonID(String personID) {
26         PersonID = personID;
27     }
28     public String getPassword() {
29         return Password;
30     }
31     public void setPassword(String password) {
32         Password = password;
33     }
34     public String getName() {
35         return Name;
36     }
37     public void setName(String name) {
38         Name = name;
39     }
40     public String getContact() {
41         return Contact;
42     }
43     public void setContact(String contact) {
44         Contact = contact;
45     }
46     public String getRole() {
47         return Role;
48     }
49     public void setRole(String role) {
50         Role = role;
51     }
```







```

52  @Override
53  public void add(Person person) {
54      // TODO Auto-generated method stub
55      try {
56          Connection con = Gallerydb.getConnection();
57          String sql = "INSERT INTO person(personid,password,name,contact,role) VALUES (?, ?, ?, ?, ?)";
58          PreparedStatement ps = con.prepareStatement(sql);
59          ps.setString(1, person.getPersonID());
60          ps.setString(2, person.getPassword());
61          ps.setString(3, person.getName());
62          ps.setString(4, person.getContact());
63          //ps.setString(5, person.getRole());
64          ps.executeUpdate();
65          JOptionPane.showMessageDialog(null, "Customer Registration Completed! :)");
66      } catch (Exception e) {
67          e.printStackTrace();
68          JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
69      }
70  }
71  }
72  @Override
73  public void update(Person person) {
74      // TODO Auto-generated method stub
75      try {
76
77          Connection con = Gallerydb.getConnection();
78          String sql = "UPDATE person SET password=?,name=?,contact=? WHERE personid=?";
79          PreparedStatement ps = con.prepareStatement(sql);
80          ps.setString(1, person.getPassword());
81          ps.setString(2, person.getName());
82          ps.setString(3, person.getContact());
83          ps.setString(4, person.getPersonID());
84          ps.executeUpdate();
85
86
87          JOptionPane.showMessageDialog(null, "Customer Detail Updated! :)");
88      } catch (Exception e) {
89          e.printStackTrace();
90          JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
91      }
92  }

```

### 3) Interface

Located Here:

- ▼  database.interfacing
  - >  ArtInterface.java
  - >  ArtistInterface.java
  - >  PersonInterface.java
  - >  ReserveInterface.java
  - >  ZoneInterface.java

```
1 //interface for Art
2 package database.interfacing;
3
4+ import java.util.List;
5
6
7
8
9
10 public interface ArtInterface {
11
12     public void add (Art art);
13     public void update (Art art);
14     public Art get (String id);
15     public List<Art> list();
16     public List<Art> list(String sql);
17 }
18
19
```

```
1 //interface for Artist
2 package database.interfacing;
3
4+ import java.util.List;
5
6
7
8 public interface ArtistInterface {
9
10     public void add (Artist artist);
11     public void update (Artist artist);
12     public Artist get (String id);
13     public List<Artist> list();
14 }
15
16
```

```

1 //interface for Person
2 package database.interfacing;
3
4*import java.util.List;
5
6
7
8 public interface PersonInterface {
9
10     public void add (Person person);
11     public void update (Person person);
12     public Person get (String id);
13     public List<Person> list();
14 }

```

```

1 //interface for Reserve
2 package database.interfacing;
3
4*import java.util.List;
5
6
7
8 public interface ReserveInterface {
9
10     public void add (Reserve reserve);
11     public void delete (Reserve reserve);
12     public void update (Reserve reserve);
13     public List<Reserve> list();
14     public List<Reserve> list(String sql);
15 }
16

```

```

1 package database.interfacing;
2
3*import java.util.List;
4
5
6
7 public interface ZoneInterface {
8
9     public Zone get (String id);
10     public List<Zone> list();
11 }
12
13

```

## 4) Java Collection:[List / Set / Map]

Located Here:

```
database.lib
> Art.java
> Artist.java
> Person.java
> Reserve.java
> Zone.java
```

### Art.java

```
140 @Override
141 public List<Art> list() {
142     // TODO Auto-generated method stub
143     List<Art> list = new ArrayList<Art>();
144     try {
145         Connection con = Gallerydb.getConnection();
146         String sql = "SELECT * FROM art ";
147         PreparedStatement ps = con.prepareStatement(sql);
148         ResultSet rs = ps.executeQuery();
149
150         while(rs.next()){
151             Art art = new Art();
152             art.setArtID(rs.getString("artid"));
153             art.setName(rs.getString("name"));
154             art.setDesc(rs.getString("description"));
155             art.setPrice(rs.getDouble("price"));
156             art.setPublishDate(rs.getString("publishdate"));
157             art.setArtistID(rs.getString("artistid"));
158             art.setZoneID(rs.getString("zoneid"));
159
160             list.add(art);
161         }
162
163     } catch (Exception e) {
164         e.printStackTrace();
165         JOptionPane.showMessageDialog(null, "Error");
166     }
167     return list;
168 }
```

## Artist.java

```
111 @Override
112 public List<Artist> list() {
113     // TODO Auto-generated method stub
114     List<Artist> list = new ArrayList<Artist>();
115     try {
116         Connection con = Gallerydb.getConnection();
117         String sql = "SELECT * FROM artist ";
118         PreparedStatement ps = con.prepareStatement(sql);
119         ResultSet rs = ps.executeQuery();
120
121         while(rs.next()){
122             Artist artist = new Artist();
123             artist.setArtistID(rs.getString("artistid"));
124             artist.setName(rs.getString("name"));
125             artist.setCountry(rs.getString("country"));
126             artist.setContact(rs.getString("contact"));
127
128             list.add(artist);
129         }
130
131     } catch (Exception e) {
132         e.printStackTrace();
133         JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
134     }
135     return list;
136 }
```

## Person.java

```
119     public List<Person> list() {
120         // TODO Auto-generated method stub
121         List<Person> list = new ArrayList<Person>();
122         try {
123             Connection con = Gallerydb.getConnection();
124             String sql = "SELECT * FROM person WHERE role = 'customer'";
125             PreparedStatement ps = con.prepareStatement(sql);
126             ResultSet rs = ps.executeQuery();
127
128             while(rs.next()){
129                 Person person = new Person();
130                 person.setPersonID(rs.getString("personid"));
131                 person.setPassword(rs.getString("password"));
132                 person.setName(rs.getString("name"));
133                 person.setContact(rs.getString("contact"));
134
135                 list.add(person);
136             }
137
138         } catch (Exception e) {
139             e.printStackTrace();
140             JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
141         }
142         return list;
143     }
144 }
```

## Reserve.java

```
120@ @Override
121 public List<Reserve> list(String sql) {
122     // TODO Auto-generated method stub
123     List<Reserve> list = new ArrayList<Reserve>();
124     try {
125         Connection con = Gallerydb.getConnection();
126         //String sql = "SELECT * FROM reserve WHERE status='PENDING'";
127         PreparedStatement ps = con.prepareStatement(sql);
128         ResultSet rs = ps.executeQuery();
129
130         while(rs.next()){
131             Reserve reserve = new Reserve();
132             reserve.setPersonID(rs.getString("personid"));
133             reserve.setArtID(rs.getString("artid"));
134             reserve.setDate(rs.getString("date"));
135             reserve.setStatus(rs.getString("status"));
136
137             list.add(reserve);
138         }
139
140     } catch (Exception e) {
141         e.printStackTrace();
142         JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
143     }
144     return list;
145 }
146 }
```

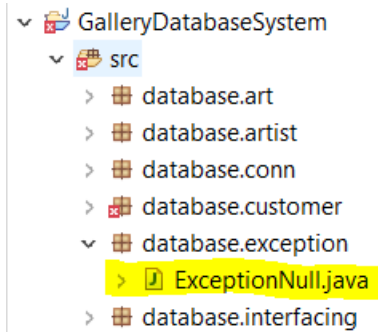


## Zone.java

```
54 @Override
55 public List<Zone> list() {
56     // TODO Auto-generated method stub
57     List<Zone> list = new ArrayList<Zone>();
58     try {
59         Connection con = Gallerydb.getConnection();
60         String sql = "SELECT * FROM zone ";
61         PreparedStatement ps = con.prepareStatement(sql);
62         ResultSet rs = ps.executeQuery();
63
64         while(rs.next()){
65             Zone zone = new Zone();
66             zone.setZoneID(rs.getString("zoneid"));
67             zone.setName(rs.getString("name"));
68
69             list.add(zone);
70         }
71     } catch (Exception e) {
72         e.printStackTrace();
73         JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
74     }
75     return list;
76 }
77
78 }
```

## 5) Exception Handling:

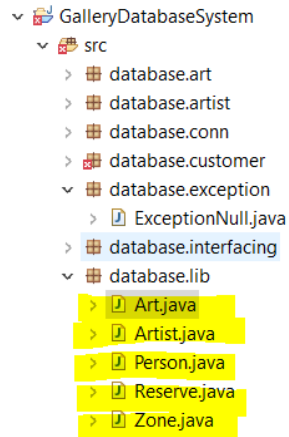
Located Here:



Code Snippet:

```
1 package database.exception;
2
3 import javax.swing.JOptionPane;
4
5 public class ExceptionNull extends Exception{
6
7     //Deserialization uses this number to ensure that a
8     //loaded class corresponds exactly to a serialized object
9     static final long serialVersionUID = 1L;
10
11     public ExceptionNull(String message){
12         super(message);
13         JOptionPane.showMessageDialog(null, message);
14     }
15
16 }
```

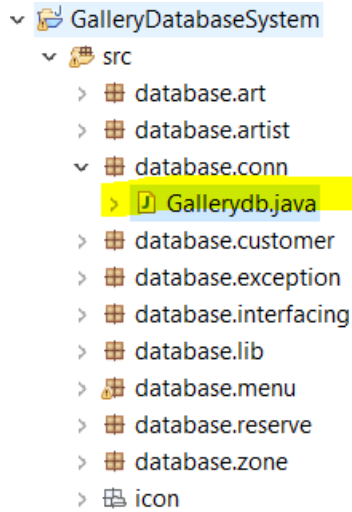
## Located Here:



## Code Snippet Template(Art.java)

```
67 @Override
68 public void add(Art art) {
69     // TODO Auto-generated method stub
70     try {
71         Connection con = Gallerydb.getConnection();
72         String sql = "INSERT INTO art(artid,name,description,publishdate,price,artistid,zoneid) VALUES(?,?,?,?,?,?,?)";
73         PreparedStatement ps = con.prepareStatement(sql);
74         ps.setString(1, art.getArtID());
75         ps.setString(2, art.getName());
76         ps.setString(3, art.getDesc());
77         ps.setString(4, art.getPublishDate());
78         ps.setDouble(5, art.getPrice());
79         ps.setString(6, art.getArtistID());
80         ps.setString(7, art.getZoneID());
81         ps.executeUpdate();
82         JOptionPane.showMessageDialog(null, "New Art Added! :");
83     } catch (Exception e) {
84         e.printStackTrace();
85         JOptionPane.showMessageDialog(null, "Oh No, There's Problem! :(");
86     }
87 }
```

## 6) Connecting to database



```
2+ * Class used to establish stable connection[]
5 package database.conn;
6
7+ import java.sql.Connection;[]
9
10 public class Gallerydb {
11
12     static Connection con;
13     static String driver = "com.mysql.jdbc.Driver";
14     static String url = "jdbc:mysql://localhost/art"; //database name
15     static String username = "root"; //root user access allow modification
16     static String password = "";
17
18
19+ public static Connection getConnection() throws Exception{
20     if(con == null){
21         Class.forName(driver);
22         con = DriverManager.getConnection(url,username, password);
23     }
24     return con;
25 }
26 }
27
28
```