Case Study: Virtual Art Gallery

Instructions

- Project submissions should **be a single zip file containing source code and an additional document with output screenshot**
- Each section builds upon the previous one, and by the end, you will have a comprehensive **Virtual Art Gallery** implemented with a strong focus on **SQL schema design**, **control flow statements, loops, arrays, collections, exception handling** ,**database interaction** using **JDBC** and **Unit Testing** using **JUnit**
- Remember to provide clear instructions, sample code, and any necessary resources or databases for each assignment.
- Follow **object-oriented principles** throughout the project. Use classes and objects to model real-world entities, **encapsulate data and behavior**, and **ensure code reusability**.
- Throw **user defined exceptions** from corresponding methods and handle in the main method**.**

1: Problem Statement:

Virtual Art Gallery System using Core Java with JDBC

Introduction The Virtual Art Gallery System aims to provide an immersive and interactive experience for art enthusiasts to explore, view, and appreciate a diverse collection of artworks online. To develop a robust and maintainable system, it is essential to create a well-structured object-oriented programming model with classes that represent the core entities and functionalities of the virtual art gallery. This problem statement outlines the requirements for designing and implementing the OOP model classes for this system.

1. Schema design:

Entities:

- Designing the schema for a Virtual Art Gallery involves creating a structured representation of the database that will store information about artworks, artists, users, galleries, and various relationships between them. Below is a schema design for a Virtual Art Gallery database:
- **Entities and Attributes:**
- **Artwork**

ArtworkID (Primary Key)

Title

Description

CreationDate

Medium

ImageURL (or any reference to the digital representation)

- **Artist**

ArtistID (Primary Key)

Name

Biography

BirthDate

Nationality

Website

Contact Information

- **User**

UserID (Primary Key)

Username

Password

Email

First Name

Last Name

Date of Birth

Profile Picture

FavoriteArtworks (a list of references to ArtworkIDs)

- **Gallery**

GalleryID (Primary Key)

Name

Description

Location

Curator (Reference to ArtistID)

OpeningHours

- **Relationships:**
- **Artwork - Artist (Many-to-One)**

An artwork is created by one artist.

Artwork.ArtistID (Foreign Key) references Artist.ArtistID.

- **User - Favorite Artwork (Many-to-Many)**

A user can have many favorite artworks, and an artwork can be a favorite of multiple users.

User_Favorite_Artwork (junction table):

UserID (Foreign Key) references User.UserID.

ArtworkID (Foreign Key) references Artwork.ArtworkID.

- **Artist - Gallery (One-to-Many)**

An artist can be associated with multiple galleries, but a gallery can have only one curator (artist).

Gallery.ArtistID (Foreign Key) references Artist.ArtistID.

- **Artwork - Gallery (Many-to-Many)**

An artwork can be displayed in multiple galleries, and a gallery can have multiple artworks.

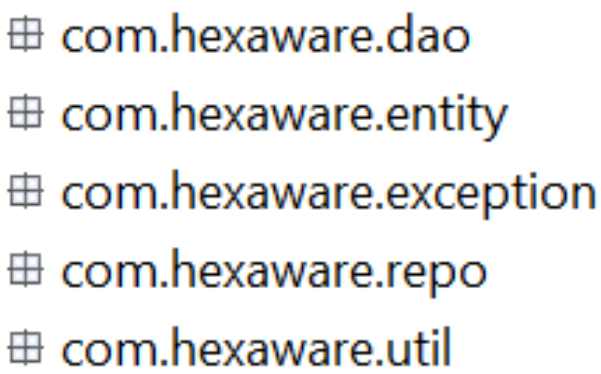
Artwork_Gallery (junction table):

ArtworkID (Foreign Key) references Artwork.ArtworkID.

GalleryID (Foreign Key) references Gallery.GalleryID.

.2: Java Project: Model classes

- The following **package structure** to be followed in the **demo application.**



- **com.hexaware.entity**
  - Create entity classes in this package. All entity class should not have any business logic.
- **com.hexaware.dao**
  - Create Service Provider interface to showcase functionalities

- Create the implementation class for the above interface with db interaction.

- **com.hexaware.exception**
  - Create user defined exceptions in this package and handle exceptions whenever needed.
- **com.hexaware.util**

  - Create a **DBPropertyUtil** class with a static function which Takes property file name as parameter and returns connection string.
  - Create a **DBConnUtil** class which holds **static method** which takes connection string as parameter file and returns **connection object(Use method defined in DBPropertyUtil class to get the connection String)**.Top of Form

Scope : Entity classes/Models/POJO, Abstraction/Encapsulation

Create the model**/entity classes corresponding to the schema** within package **com.hexaware.entities** with variables declared private, constructors(default and parametrized,getters,setters and toString())

6: Service Provider Interface

Keep the interfaces and implementation classes in package com.hexaware.dao

Create **IVirtualArtGallery** Interface

```
public interface IVirtualArtGallery {
    // Artwork Management
    boolean addArtwork(Artwork artwork);
    boolean updateArtwork(Artwork artwork);
```

```
    boolean removeArtwork(int artworkId);
    Artwork getArtworkById(int artworkId);
    List<Artwork> searchArtworks(String keyword);


    // User Favorites
    boolean addArtworkToFavorite(int userId, int artworkId);
    boolean removeArtworkFromFavorite(int userId, int artworkId);
    List<Artwork> getUserFavoriteArtworks(int userId);
}
```

7: JDBC and Database Interaction

Scope: Connection class, Resultset, execute(),executeQuery(),SQL exception, datatype compatibility

Task: Connect your application to the SQL database **:**

1. Write code to establish a connection to your SQL database.

   Create a utility class **DBConnection** in a package **com.hexaware.util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

   Connection properties supplied in the connection string should be read from a property file.

   Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

7: Service implementation

(Scope: Interfaces, Implementation classes, Object Arrays, Collections)

1. Create a Service class **CrimeAnalysisServiceImpl** in **com.hexaware.dao** with a static variable named connection of type **Connection** which can be assigned in the constructor by invoking the **getConnection()** method in **DBConnection** class

2. Provide implementation for all the methods in the interface by using jdbc.

8: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,thow & throws keyword usage)

Create the exceptions in package **com.hexaware.myexceptions**

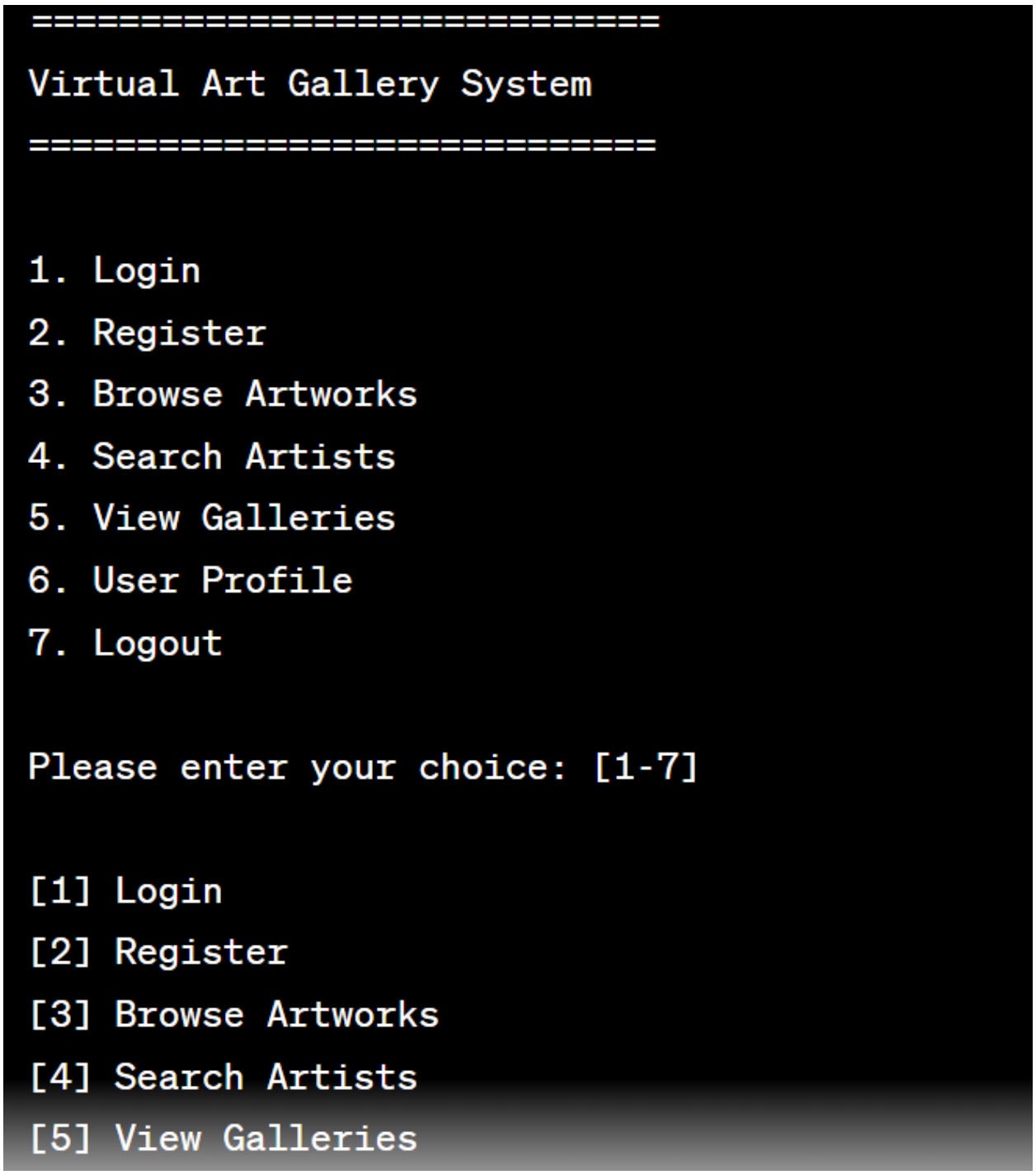
Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. **ArtWorkNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db
2. **UserNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db

9. Main Method

Create class named MainModule with main method in com.hexware.mainmod.

Trigger all the methods in service implementation class.



10. Unit Testing

Creating JUnit test cases for a Virtual Art Gallery system is essential to ensure that the system functions correctly. Below are sample test case questions that can serve as a starting point for your JUnit test suite:

1. User Management:

a. Test whether a new user can successfully register in the system. b. Verify that an existing user cannot register with the same username. c. Test the login process for a registered user. d. Check if an unregistered user fails to log in.

2. Artwork Management:

a. Test the ability to upload a new artwork to the gallery.

b. Verify that updating artwork details works correctly.

c. Test removing an artwork from the gallery.

d. Check if searching for artworks returns the expected results.

3. Artist Management:

a. Test adding a new artist to the system.

b. Verify that updating artist information is successful.

c. Test removing an artist from the system.

d. Check if searching for artists returns the expected results.

4. Gallery Management:

a. Test creating a new gallery.

b. Verify that updating gallery information works correctly.

c. Test removing a gallery from the system.

d. Check if searching for galleries returns the expected results.

5. Error Handling:

a. Test error handling for invalid inputs during user registration.

b. Verify that errors are handled appropriately during the login process.

c. Test handling exceptions when trying to access non-existent galleries or artworks.