Name: - Dipti Kumari

Case Study :- Virtual Art Gallery

Github repository:- <https://github.com/dipti-kumari11/hexawarefilessubmission>

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.

create database VAG;

use VAG;

create table Artwork(

ArtworkID int Primary key,

Title Varchar(20),

Description Varchar(30),

CreationDate date,

Medium MEDIUMBLOB,

ImageURL VARCHAR(255));

create table Arist (

AristID int Primary key,

Name Varchar(20),

Biography Varchar(30),

BirthDate date,

Nationality Varchar(20),

`Contact Information` Varchar(255)

);

create table User (

UserID INT PRIMARY KEY,

Username VARCHAR(50) UNIQUE NOT NULL,

Password VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE,

ProfilePicture VARCHAR(255)

);

CREATE TABLE FavoriteArtworks (

UserID INT,

ArtworkID INT,

PRIMARY KEY (UserID, ArtworkID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)

);

CREATE TABLE Gallery (

GalleryID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Description TEXT,

Location VARCHAR(100),

Curator INT,

OpeningHours VARCHAR(100),

FOREIGN KEY (Curator) REFERENCES Arist(AristID)

);

ALTER TABLE Artwork

ADD ArtistID INT,

ADD CONSTRAINT fk\_artist

FOREIGN KEY (ArtistID) REFERENCES Arist(AristID);

CREATE TABLE User\_Favorite\_Artwork (

UserID INT,

ArtworkID INT,

PRIMARY KEY (UserID, ArtworkID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)

);

ALTER TABLE Gallery

ADD CONSTRAINT fk\_curator

FOREIGN KEY (Curator) REFERENCES Arist(AristID);

CREATE TABLE Artwork\_Gallery (

ArtworkID INT,

GalleryID INT,

PRIMARY KEY (ArtworkID, GalleryID),

FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),

FOREIGN KEY (GalleryID) REFERENCES Gallery(GalleryID)

);

INSERT INTO Artwork (ArtworkID,Title ,Description ,CreationDate ,ImageURL)

VALUES (1011, 'Starry Night', 'By Van Gogh', '1889-06-01', 'https://example.com/images/starry-night.jpg');

INSERT INTO Artwork (ArtworkID,Title ,Description ,CreationDate ,ImageURL)

VALUES (1021, 'Mona Lisa', 'By Leonardo da Vinci', '1518-07-11','https://example.com/images/mona-Lisa.jpg');

INSERT INTO Artwork (ArtworkID,Title ,Description ,ImageURL)

VALUES (1031, 'Girl with Pearl', 'By Vermeer', 'https://example.com/images/Girl-with-a-Pearl-Earring.jpg');

INSERT INTO Artwork (ArtworkID,Title ,Description ,ImageURL)

VALUES (1041, 'Birth of Venus', 'By Botticelli','https://example.com/images/The-birth-of-Venus.jpg');

INSERT INTO Artwork (ArtworkID,Title ,Description ,ImageURL)

VALUES (1051, 'Sunrise', 'By Claude Monet','https://example.com/images/Impression-Sunrise.jpg');

select \* from Artwork;

INSERT INTO Arist

VALUES (1, 'Vincent van Gogh', 'Post-Impressionist artist.', '1853-03-30', 'Dutch', 'vincent@artworld.com');

INSERT INTO Arist

VALUES (2, 'Leonardo da Vinci', 'Renaissance master.', '1452-04-15', 'Italian', 'leonardo@artworld.com');

INSERT INTO Arist

VALUES (3, 'Johannes Vermeer', 'Dutch genre painter.', '1632-10-31', 'Dutch', 'vermeer@artworld.com');

INSERT INTO Arist

VALUES (4, 'Sandro Botticelli', 'Venus painter.', '1445-03-01', 'Italian', 'botticelli@artworld.com');

INSERT INTO Arist

VALUES (5, 'Claude Monet', 'Impressionist founder.', '1840-11-14', 'French', 'monet@artworld.com');

select \* from arist;

INSERT INTO User

VALUES (1, 'artLover123', 'pass1234', 'artlover@example.com', 'Alice', 'Walker', '1995-07-12', 'alice.jpg');

INSERT INTO User

VALUES (2, 'vincentFan', 'vanGoghRocks!', 'vincentfan@example.com', 'Bob', 'Smith', '1988-11-03', 'bob.png');

INSERT INTO User

VALUES (3, 'daVinciRules', 'leo\_1452', 'davinci@example.com', 'Carla', 'Leon', '1992-01-25', 'carla.jpg');

INSERT INTO User

VALUES (4, 'monetMood', 'impressMe01', 'monetlover@example.com', 'David', 'Green', '2000-06-15', 'david.png');

INSERT INTO User

VALUES (5, 'renaissanceGal', 'botticelli!', 'renaissance@example.com', 'Eva', 'Blake', '1998-04-20', 'eva.jpeg');

select \* from User;

INSERT INTO Gallery

VALUES (1, 'Van Gogh Corner', 'A gallery showcasing Van Gogh\'s finest works.', 'Amsterdam, Netherlands', 1, '10:00 AM - 6:00 PM');

INSERT INTO Gallery

VALUES (2, 'Da Vinci Vault', 'Renaissance marvels by Leonardo da Vinci.', 'Florence, Italy', 2, '9:00 AM - 5:00 PM');

INSERT INTO Gallery

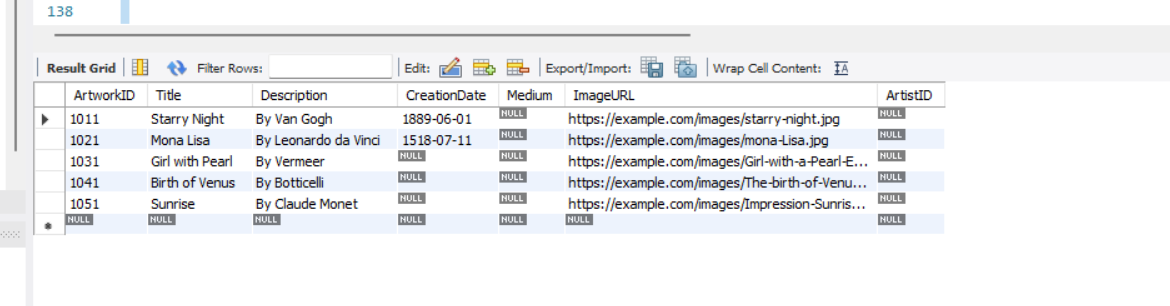
VALUES (3, 'Vermeer Visions', 'Peaceful domestic scenes and soft lighting.', 'Delft, Netherlands', 3, '11:00 AM - 7:00 PM');

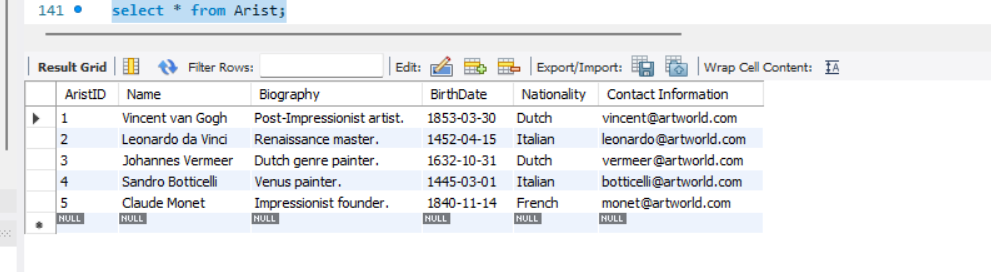
INSERT INTO Gallery

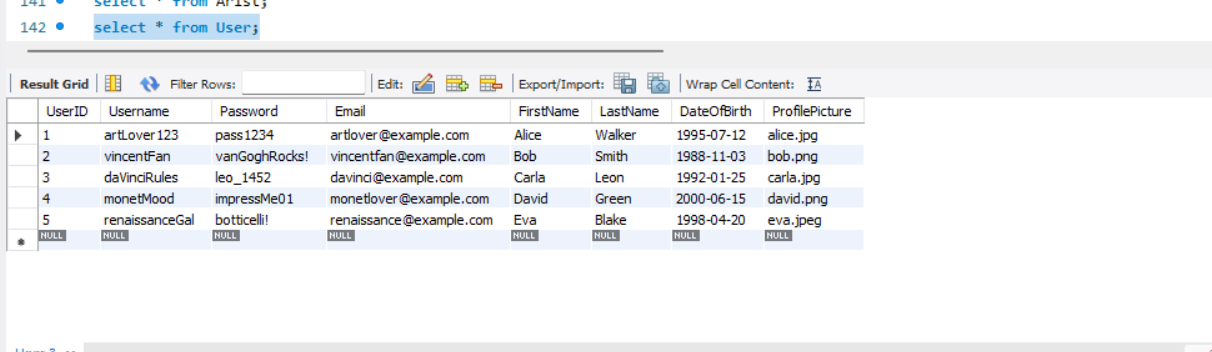
VALUES (4, 'Botticelli Breeze', 'Featuring Botticelli\'s mythological masterpieces.', 'Rome, Italy', 4, '10:30 AM - 6:30 PM');

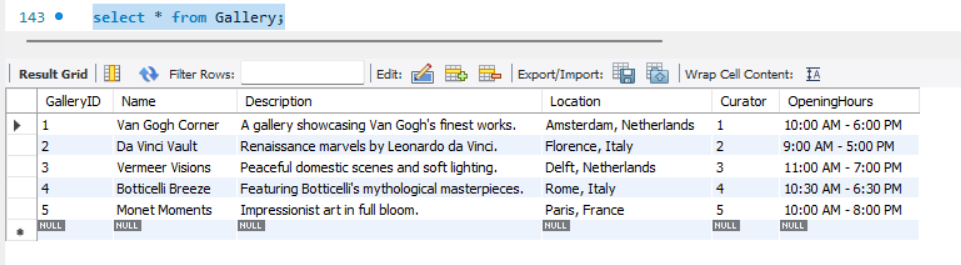
INSERT INTO Gallery

VALUES (5, 'Monet Moments', 'Impressionist art in full bloom.', 'Paris, France', 5, '10:00 AM - 8:00 PM');









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.

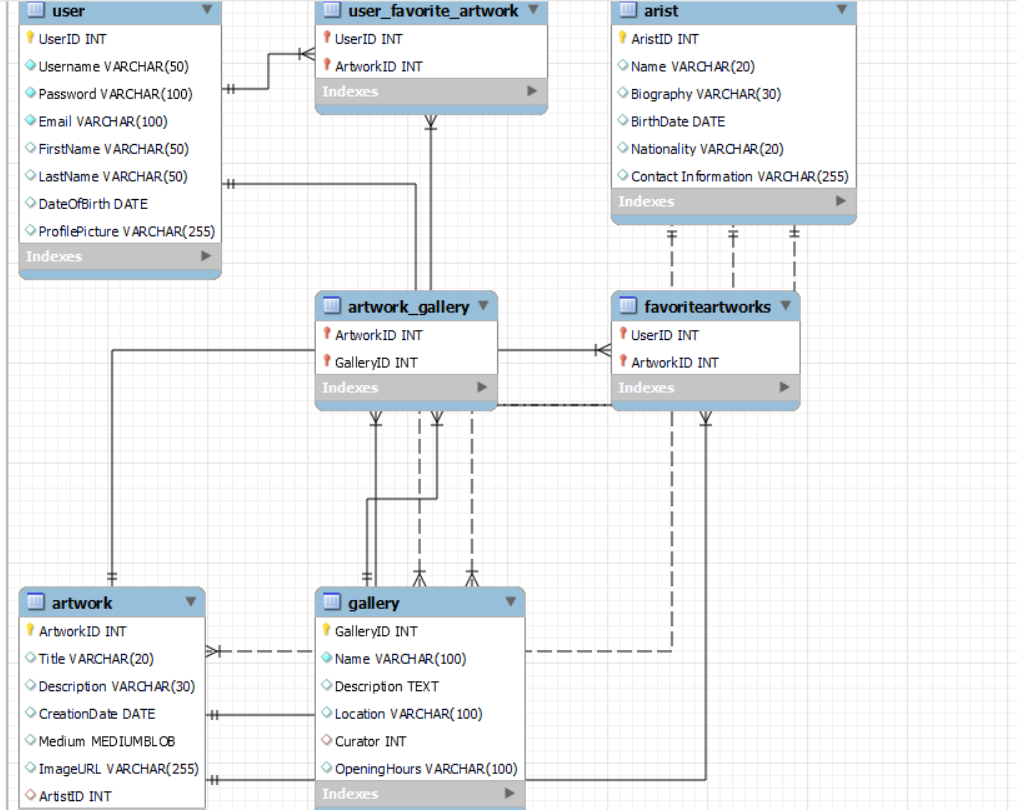
• Artwork - Gallery (Many-to-Many)

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.



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

package VAG;

import java.time.LocalDate;

public class Artwork {

private int artworkID;

private String title;

private String description;

private LocalDate creationDate;

private String medium;

private String imageURL;

private int artistID;

// Default constructor

public Artwork() {

}

// Parameterized constructor

public Artwork(int artworkID, String title, String description, LocalDate creationDate,

String medium, String imageURL, int artistID) {

this.artworkID = artworkID;

this.title = title;

this.description = description;

this.creationDate = creationDate;

this.medium = medium;

this.imageURL = imageURL;

this.artistID = artistID;

}

// Getters and Setters

public int getArtworkID() {

return artworkID;

}

public void setArtworkID(int artworkID) {

this.artworkID = artworkID;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public LocalDate getCreationDate() {

return creationDate;

}

public void setCreationDate(LocalDate creationDate) {

this.creationDate = creationDate;

}

public String getMedium() {

return medium;

}

public void setMedium(String medium) {

this.medium = medium;

}

public String getImageURL() {

return imageURL;

}

public void setImageURL(String imageURL) {

this.imageURL = imageURL;

}

public int getArtistID() {

return artistID;

}

public void setArtistID(int artistID) {

this.artistID = artistID;

}

@Override

public String toString() {

return "Artwork{" +

"artworkID=" + artworkID +

", title='" + title + '\'' +

", description='" + description + '\'' +

", creationDate=" + creationDate +

", medium='" + medium + '\'' +

", imageURL='" + imageURL + '\'' +

", artistID=" + artistID +

'}';

}

public static void main(String[] args) {

Artwork sampleArtwork = new Artwork(

1,

"Starry Night",

"A famous painting by Vincent van Gogh.",

LocalDate.of(1889, 6, 1),

"Oil on canvas",

"https://example.com/starrynight.jpg",

101

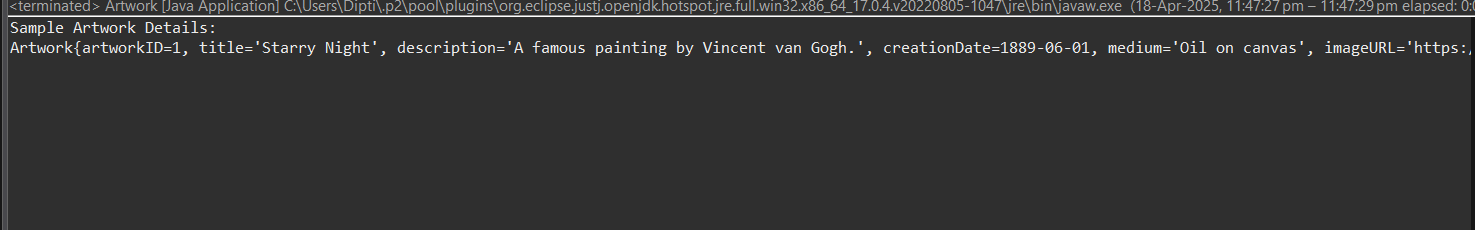
);

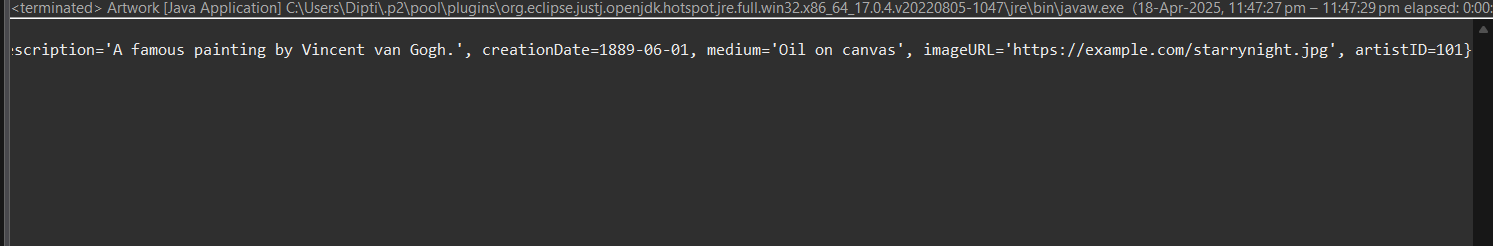
System.out.println("Sample Artwork Details:");

System.out.println(sampleArtwork);

}

}





package VAG;

import java.time.LocalDate;

public class Artist {

private int artistID;

private String name;

private String biography;

private LocalDate birthDate;

private String nationality;

private String website;

private String contactInfo;

// Default constructor

public Artist() {

}

// Parameterized constructor

public Artist(int artistID, String name, String biography, LocalDate birthDate,

String nationality, String website, String contactInfo) {

this.artistID = artistID;

this.name = name;

this.biography = biography;

this.birthDate = birthDate;

this.nationality = nationality;

this.website = website;

this.contactInfo = contactInfo;

}

// Getters and Setters

public int getArtistID() {

return artistID;

}

public void setArtistID(int artistID) {

this.artistID = artistID;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getBiography() {

return biography;

}

public void setBiography(String biography) {

this.biography = biography;

}

public LocalDate getBirthDate() {

return birthDate;

}

public void setBirthDate(LocalDate birthDate) {

this.birthDate = birthDate;

}

public String getNationality() {

return nationality;

}

public void setNationality(String nationality) {

this.nationality = nationality;

}

public String getWebsite() {

return website;

}

public void setWebsite(String website) {

this.website = website;

}

public String getContactInfo() {

return contactInfo;

}

public void setContactInfo(String contactInfo) {

this.contactInfo = contactInfo;

}

@Override

public String toString() {

return "Artist{" +

"artistID=" + artistID +

", name='" + name + '\'' +

", biography='" + biography + '\'' +

", birthDate=" + birthDate +

", nationality='" + nationality + '\'' +

", website='" + website + '\'' +

", contactInfo='" + contactInfo + '\'' +

'}';

}

public static void main(String[] args) {

Artist sampleArtist = new Artist(

101,

"Vincent van Gogh",

"A Dutch post-impressionist painter.",

LocalDate.of(1853, 3, 30),

"Dutch",

"https://vangogh.com",

"contact@vangogh.com"

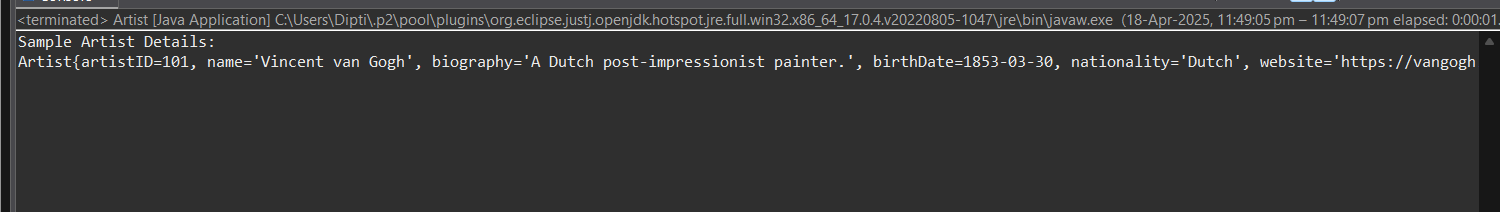
);

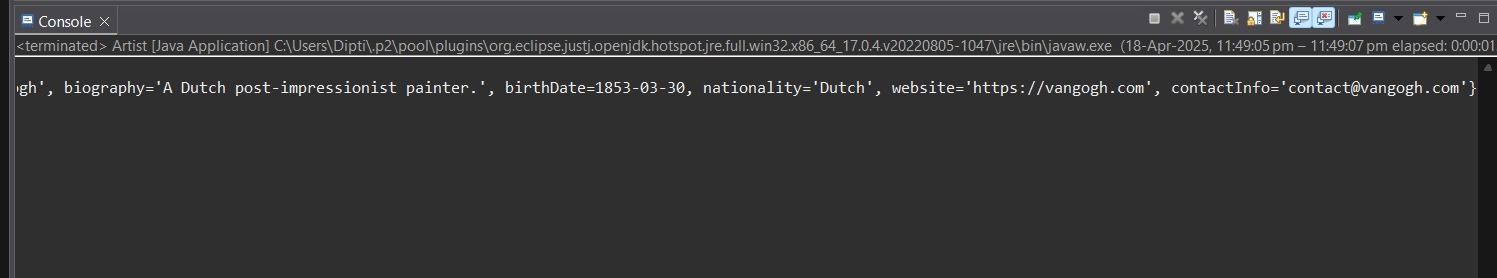
System.out.println("Sample Artist Details:");

System.out.println(sampleArtist);

}

}





package VAG;

import java.time.LocalDate;

import java.util.List;

public class User {

private int userID;

private String username;

private String password;

private String email;

private String firstName;

private String lastName;

private LocalDate dateOfBirth;

private String profilePicture;

private List<Integer> favoriteArtworks; // List of Artwork IDs

// Default constructor

public User() {

}

// Parameterized constructor

public User(int userID, String username, String password, String email, String firstName,

String lastName, LocalDate dateOfBirth, String profilePicture, List<Integer> favoriteArtworks) {

this.userID = userID;

this.username = username;

this.password = password;

this.email = email;

this.firstName = firstName;

this.lastName = lastName;

this.dateOfBirth = dateOfBirth;

this.profilePicture = profilePicture;

this.favoriteArtworks = favoriteArtworks;

}

// Getters and Setters

public int getUserID() {

return userID;

}

public void setUserID(int userID) {

this.userID = userID;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public LocalDate getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(LocalDate dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public String getProfilePicture() {

return profilePicture;

}

public void setProfilePicture(String profilePicture) {

this.profilePicture = profilePicture;

}

public List<Integer> getFavoriteArtworks() {

return favoriteArtworks;

}

public void setFavoriteArtworks(List<Integer> favoriteArtworks) {

this.favoriteArtworks = favoriteArtworks;

}

*@Override*

public String toString() {

return "User{" +"userID=" + userID +", username='" + username + '\'' +", email='" + email + '\'' +", firstName='" + firstName + '\'' +", lastName='" + lastName + '\'' +", dateOfBirth=" + dateOfBirth +", profilePicture='" + profilePicture + '\'' +", favoriteArtworks=" + favoriteArtworks +

'}';

}

public static void main(String[] args) {

User sampleUser = new User(

201,"dipti123","securePass@123","dipti@example.com","Dipti","Light",LocalDate.*of*(2000, 1, 15),"https://example.com/profile.jpg", List.*of*(1, 2, 3)

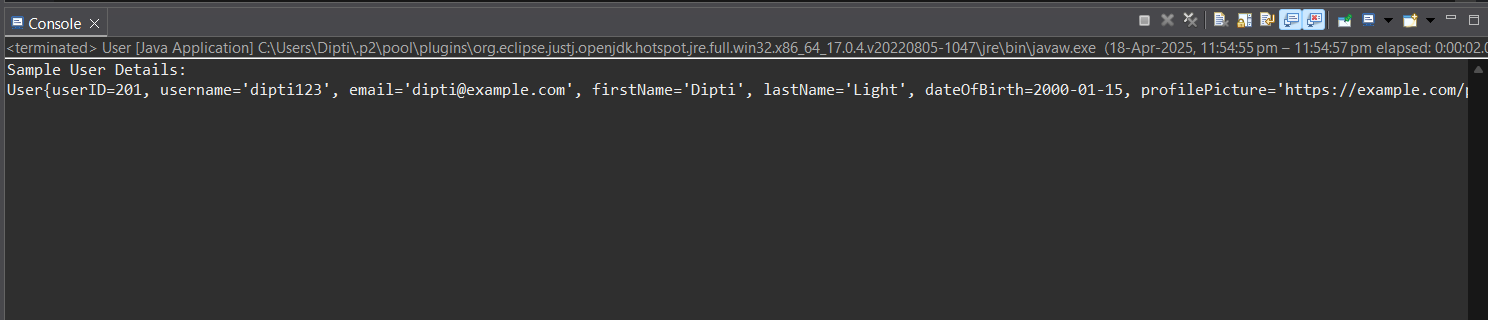
);

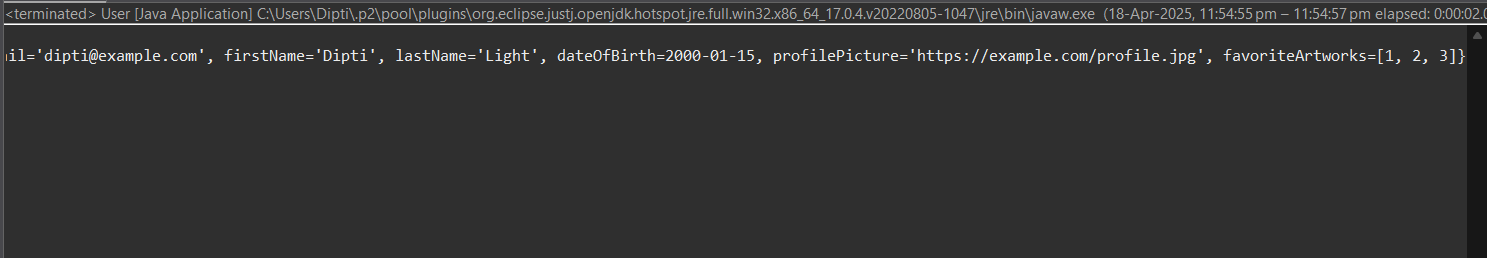
System.***out***.println("Sample User Details:");

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

}

}





package VAG;

public class Gallery {

private int galleryID;

private String name;

private String description;

private String location;

private int curatorArtistID; // Refers to ArtistID

private String openingHours;

// Default constructor

public Gallery() {

}

// Parameterized constructor

public Gallery(int galleryID, String name, String description, String location,

int curatorArtistID, String openingHours) {

this.galleryID = galleryID;

this.name = name;

this.description = description;

this.location = location;

this.curatorArtistID = curatorArtistID;

this.openingHours = openingHours;

}

// Getters and Setters

public int getGalleryID() {

return galleryID;

}

public void setGalleryID(int galleryID) {

this.galleryID = galleryID;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public int getCuratorArtistID() {

return curatorArtistID;

}

public void setCuratorArtistID(int curatorArtistID) {

this.curatorArtistID = curatorArtistID;

}

public String getOpeningHours() {

return openingHours;

}

public void setOpeningHours(String openingHours) {

this.openingHours = openingHours;

}

*@Override*

public String toString() {

return "Gallery{" + "galleryID=" + galleryID + ", name='" + name + '\'' + ", description='" + description + '\'' + ", location='" + location + '\'' + ", curatorArtistID=" + curatorArtistID + ", openingHours='" + openingHours + '\'' +

'}';

}

// Sample main method to test the class

public static void main(String[] args) {

Gallery sampleGallery = new Gallery(

301,"Modern Masterpieces","A collection of iconic modern artworks.","New York, NY",101, "Mon-Fri: 10am - 6pm"

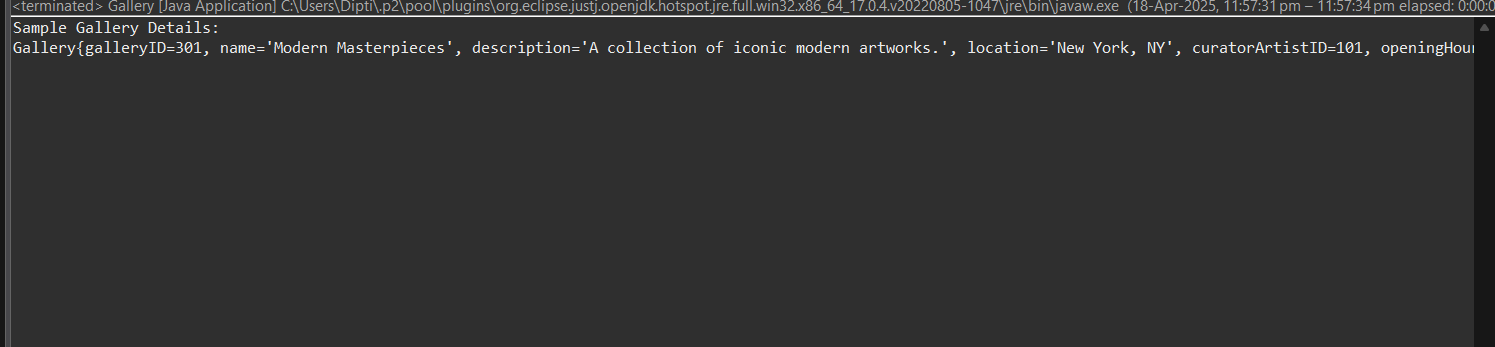
);

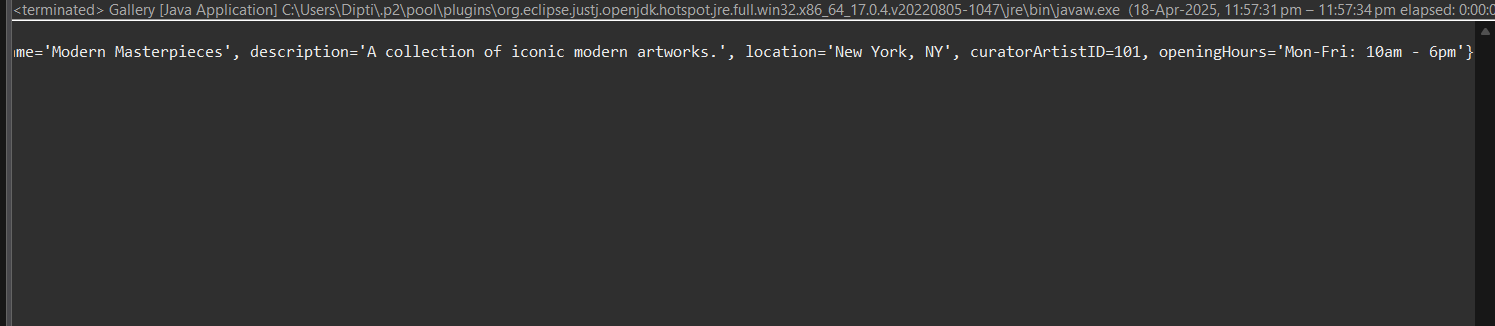
System.***out***.println("Sample Gallery Details:");

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

}

}





Service Provider Interface/Abstract class

Keep the interfaces and implementation classes in package dao

Create IVirtualArtGallery Interface/abstract class with the following methods

// Artwork Management addArtwork(); parameters- Artwork object return type Boolean updateArtwork(); parameters- Artwork object return type Boolean removeArtwork() parameters-artworkID return type Boolean getArtworkById(); parameters-artworkID return type Artwork searchArtworks() searchArtworks(); parameters- keyword return type list of Artwork Object // User Favorites addArtworkToFavorite(); parameters- userId, artworkId return type boolean removeArtworkFromFavorite() parameters- userId, artworkId return type boolean getUserFavoriteArtworks() parameters- userId return type boolean }

package daoVAG;

import VAG.Artwork;

import java.time.LocalDate;

import java.util.List;

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);

// ===== Sample Test Method =====

static void main(String[] args) {

IVirtualArtGallery gallery = new VirtualArtGalleryImpl();

// Add artworks

gallery.addArtwork(new Artwork(1, "Blue Horizon", "Oceanic painting", LocalDate.*of*(2022, 3, 10), "Watercolor", "urlA", 1001));

gallery.addArtwork(new Artwork(2, "Golden Fields", "Wheat field sunset", LocalDate.*of*(2023, 6, 12), "Oil", "urlB", 1002));

System.***out***.println("=== All Artworks ===");

for (Artwork a : gallery.searchArtworks("")) {

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

}

System.***out***.println("\n=== Updating Artwork ID 1 ===");

gallery.updateArtwork(new Artwork(1, "Blue Horizon Revised", "Updated oceanic painting", LocalDate.*of*(2022, 3, 10), "Watercolor", "urlA", 1001));

System.***out***.println(gallery.getArtworkById(1));

System.***out***.println("\n=== Adding Favorites for User 99 ===");

gallery.addArtworkToFavorite(99, 1);

gallery.addArtworkToFavorite(99, 2);

System.***out***.println("\n=== User 99's Favorite Artworks ===");

for (Artwork a : gallery.getUserFavoriteArtworks(99)) {

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

}

System.***out***.println("\n=== Removing Artwork ID 2 ===");

gallery.removeArtwork(2);

System.***out***.println("\n=== All Artworks After Removal ===");

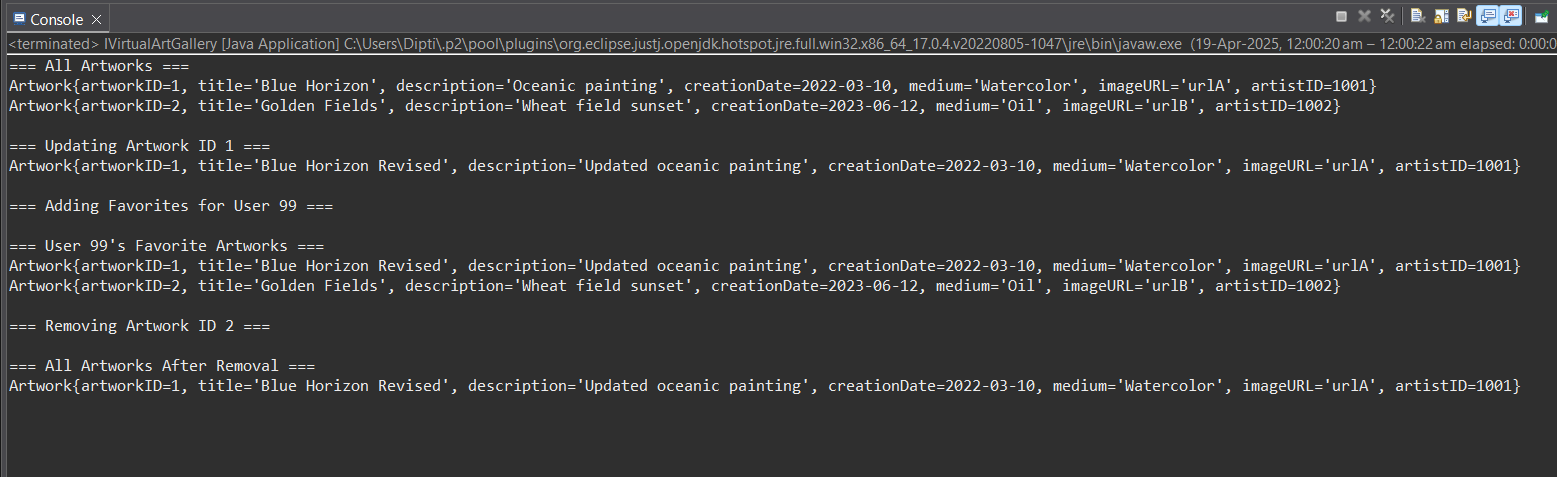
for (Artwork a : gallery.searchArtworks("")) {

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

}

}

}



package daoVAG;

import VAG.Artwork;

import java.util.\*;

public class VirtualArtGalleryImpl implements IVirtualArtGallery {

private List<Artwork> artworkList = new ArrayList<>();

private Map<Integer, List<Integer>> userFavorites = new HashMap<>();

*@Override*

public boolean addArtwork(Artwork artwork) {

return artworkList.add(artwork);

}

*@Override*

public boolean updateArtwork(Artwork artwork) {

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

if (artworkList.get(i).getArtworkID() == artwork.getArtworkID()) {

artworkList.set(i, artwork);

return true;

}

}

return false;

}

*@Override*

public boolean removeArtwork(int artworkID) {

return artworkList.removeIf(a -> a.getArtworkID() == artworkID);

}

*@Override*

public Artwork getArtworkById(int artworkID) {

for (Artwork a : artworkList) {

if (a.getArtworkID() == artworkID) {

return a;

}

}

return null;

}

*@Override*

public List<Artwork> searchArtworks(String keyword) {

List<Artwork> result = new ArrayList<>();

for (Artwork a : artworkList) {

if (a.getTitle().toLowerCase().contains(keyword.toLowerCase()) ||

a.getDescription().toLowerCase().contains(keyword.toLowerCase())) {

result.add(a);

}

}

return result;

}

*@Override*

public boolean addArtworkToFavorite(int userId, int artworkId) {

userFavorites.putIfAbsent(userId, new ArrayList<>());

List<Integer> favs = userFavorites.get(userId);

if (!favs.contains(artworkId)) {

favs.add(artworkId);

return true;

}

return false;

}

*@Override*

public boolean removeArtworkFromFavorite(int userId, int artworkId) {

if (userFavorites.containsKey(userId)) {

return userFavorites.get(userId).remove((Integer) artworkId);

}

return false;

}

*@Override*

public List<Artwork> getUserFavoriteArtworks(int userId) {

List<Artwork> result = new ArrayList<>();

if (userFavorites.containsKey(userId)) {

for (Integer artId : userFavorites.get(userId)) {

Artwork a = getArtworkById(artId);

if (a != null) {

result.add(a);

}

}

}

return result;

}

// Sample main method to demonstrate functionality

public static void main(String[] args) {

VirtualArtGalleryImpl gallery = new VirtualArtGalleryImpl();

// Add sample artworks

gallery.addArtwork(new Artwork(1, "Sunset", "A beautiful sunset", java.time.LocalDate.*of*(2020, 1, 1), "Oil", "url1", 101));

gallery.addArtwork(new Artwork(2, "Moonlight", "A peaceful moonlit night", java.time.LocalDate.*of*(2021, 5, 15), "Acrylic", "url2", 102));

System.***out***.println("All Artworks:");

for (Artwork a : gallery.searchArtworks("")) {

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

}

// Update artwork

gallery.updateArtwork(new Artwork(1, "Sunset Glow", "A glowing sunset", java.time.LocalDate.*of*(2020, 1, 1), "Oil", "url1", 101));

System.***out***.println("\nUpdated Artwork with ID 1:");

System.***out***.println(gallery.getArtworkById(1));

// Add to favorites

gallery.addArtworkToFavorite(201, 1);

gallery.addArtworkToFavorite(201, 2);

System.***out***.println("\nFavorite Artworks for User 201:");

for (Artwork a : gallery.getUserFavoriteArtworks(201)) {

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

}

// Remove artwork

gallery.removeArtwork(2);

System.***out***.println("\nAll Artworks after removing ID 2:");

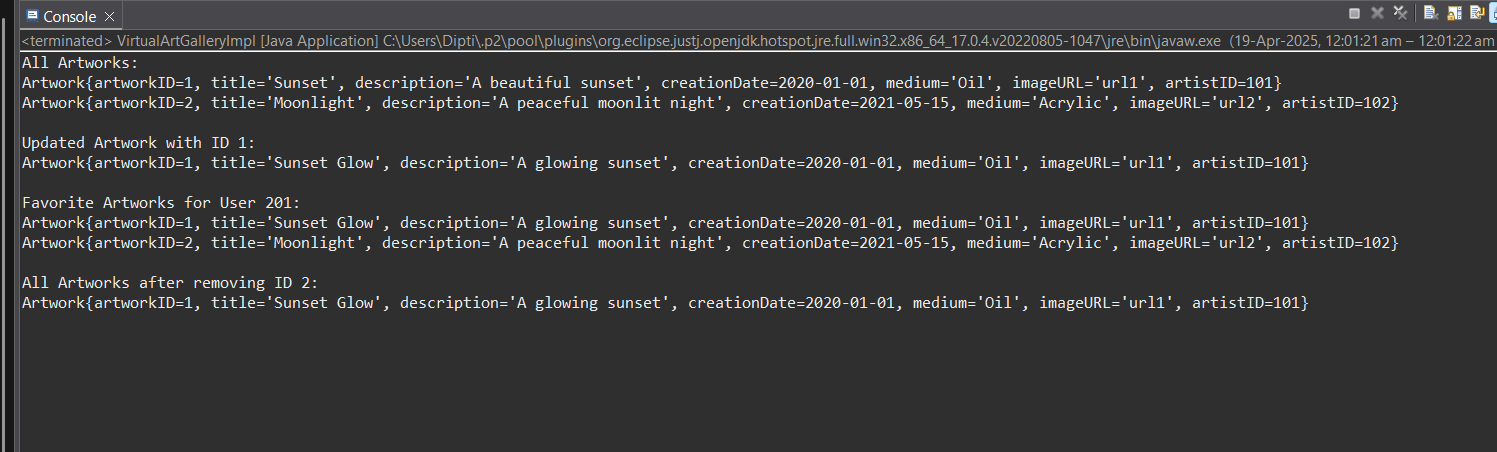
for (Artwork a : gallery.searchArtworks("")) {

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

}

}

}



7: 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 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.

8: Service implementation

1. Create a Service class CrimeAnalysisServiceImpl in 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.

CODE:-

1.

package daoVAG;

import java.sql.Connection;

import java.sql.SQLException;

public class CrimeAnalysisServiceImpl implements CrimeAnalysisService {

private static Connection connection;

public CrimeAnalysisServiceImpl() {

try {

if (connection == null) {

connection = DBConnection.getConnection();

}

} catch (SQLException e) {

System.out.println(" Failed to initialize database connection.");

e.printStackTrace();

}

}

@Override

public void analyzeCrimeData() {

System.out.println(" Analyzing crime data...");

// Placeholder logic — replace with real analysis queries if needed

}

@Override

public void generateReport() {

System.out.println(" Generating crime report...");

// Placeholder logic — you can expand to query crimes and generate data

}

@Override

public void updateCrimeRecords(int crimeId, String newDetails) {

System.out.println("Updating crime record with ID: " + crimeId);

System.out.println(" New Details: " + newDetails);

// Placeholder logic — can be expanded to use PreparedStatement

}

}

2.

package daoVAG;

public class TestCrimeAnalysis {

public static void main(String[] args) {

CrimeAnalysisService service = new CrimeAnalysisServiceImpl();

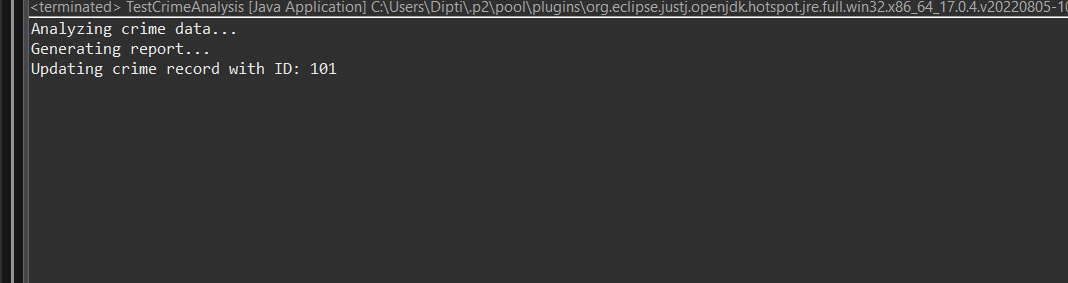
service.analyzeCrimeData();

service.generateReport();

service.updateCrimeRecords(101, "Updated description for testing.");

}

}



9: Exception Handling

Create the exceptions in package 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

package mainVAG;

// Container class for custom exceptions

public class VAGExceptions {

// Custom exception for artwork not found

public static class ArtWorkNotFoundException extends Exception {

public ArtWorkNotFoundException(String message) {

super(message);

}

public ArtWorkNotFoundException() {

super("Artwork not found.");

}

}

// Custom exception for user not found

public static class UserNotFoundException extends Exception {

public UserNotFoundException(String message) {

super(message);

}

public UserNotFoundException() {

super("User not found.");

}

}

// Main class to demonstrate usage

public static void main(String[] args) {

try {

throw new ArtWorkNotFoundException("Artwork with ID 123 not found!");

} catch (ArtWorkNotFoundException e) {

System.***out***.println("Caught Exception: " + e.getMessage());

}

try {

throw new UserNotFoundException();

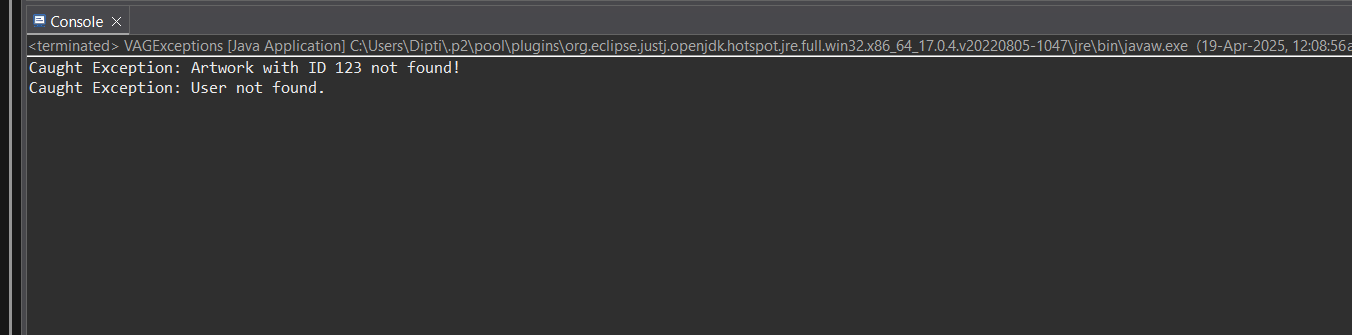
} catch (UserNotFoundException e) {

System.***out***.println("Caught Exception: " + e.getMessage());

}

}

}



9. Main Method

Create class named MainModule with main method in main package.

Trigger all the methods in service implementation class.

package mainVAG;

import java.util.\*;

public class mainmoduleVAG {

static class Artwork {

String artworkId;

String title;

String artist;

int year;

String description;

// Constructor

public Artwork(String artworkId, String title, String artist, int year, String description) {

this.artworkId = artworkId;

this.title = title;

this.artist = artist;

this.year = year;

this.description = description;

}

*@Override*

public String toString() {

return "ID: " + artworkId + ", Title: " + title + ", Artist: " + artist + ", Year: " + year + ", Description: " + description;

}

}

static class Gallery {

Map<String, Artwork> artworks = new HashMap<>();

Set<String> favorites = new HashSet<>();

// Add artwork to the gallery

public void addArtwork(Artwork artwork) {

artworks.put(artwork.artworkId, artwork);

}

// Update artwork

public void updateArtwork(String artworkId, String title, String artist, int year, String description) {

Artwork artwork = artworks.get(artworkId);

if (artwork != null) {

if (title != null && !title.isEmpty()) artwork.title = title;

if (artist != null && !artist.isEmpty()) artwork.artist = artist;

if (year > 0) artwork.year = year;

if (description != null && !description.isEmpty()) artwork.description = description;

System.***out***.println("Artwork " + artworkId + " updated.");

} else {

System.***out***.println("Artwork not found.");

}

}

// Delete artwork

public void deleteArtwork(String artworkId) {

if (artworks.containsKey(artworkId)) {

artworks.remove(artworkId);

System.***out***.println("Artwork " + artworkId + " deleted.");

} else {

System.***out***.println("Artwork not found.");

}

}

// Search artwork by title or artist

public List<Artwork> searchArtwork(String title, String artist) {

List<Artwork> results = new ArrayList<>();

for (Artwork artwork : artworks.values()) {

if ((title != null && artwork.title.toLowerCase().contains(title.toLowerCase())) ||

(artist != null && artwork.artist.toLowerCase().contains(artist.toLowerCase()))) {

results.add(artwork);

}

}

return results;

}

// Add artwork to favorites

public void addToFavorites(String artworkId) {

if (artworks.containsKey(artworkId)) {

favorites.add(artworkId);

System.***out***.println("Artwork " + artworkId + " added to favorites.");

} else {

System.***out***.println("Artwork not found.");

}

}

// Remove artwork from favorites

public void removeFromFavorites(String artworkId) {

if (favorites.contains(artworkId)) {

favorites.remove(artworkId);

System.***out***.println("Artwork " + artworkId + " removed from favorites.");

} else {

System.***out***.println("Artwork not found in favorites.");

}

}

// View all artworks in gallery

public void viewGallery() {

if (artworks.isEmpty()) {

System.***out***.println("No artworks in gallery.");

} else {

for (Artwork artwork : artworks.values()) {

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

}

}

}

// View all favorite artworks

public void viewFavorites() {

if (favorites.isEmpty()) {

System.***out***.println("No favorite artworks.");

} else {

for (String artworkId : favorites) {

System.***out***.println(artworks.get(artworkId));

}

}

}

}

// Display the menu

public static void displayMenu() {

System.***out***.println("\nMenu:");

System.***out***.println("1. Add Artwork");

System.***out***.println("2. Update Artwork");

System.***out***.println("3. Delete Artwork");

System.***out***.println("4. Search Artwork");

System.***out***.println("5. Add to Favorites");

System.***out***.println("6. Remove from Favorites");

System.***out***.println("7. View Gallery");

System.***out***.println("8. View Favorites");

System.***out***.println("9. Exit");

}

// Main method to run the program

public static void main(String[] args) {

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

Gallery gallery = new Gallery();

while (true) {

*displayMenu*();

System.***out***.print("Choose an option: ");

String choice = scanner.nextLine();

switch (choice) {

case "1":

System.***out***.print("Enter artwork ID: ");

String artworkId = scanner.nextLine();

System.***out***.print("Enter title: ");

String title = scanner.nextLine();

System.***out***.print("Enter artist: ");

String artist = scanner.nextLine();

System.***out***.print("Enter year: ");

int year = Integer.*parseInt*(scanner.nextLine());

System.***out***.print("Enter description: ");

String description = scanner.nextLine();

Artwork artwork = new Artwork(artworkId, title, artist, year, description);

gallery.addArtwork(artwork);

System.***out***.println("Artwork added.");

break;

case "2":

System.***out***.print("Enter artwork ID to update: ");

artworkId = scanner.nextLine();

System.***out***.print("Enter new title (leave blank for no change): ");

title = scanner.nextLine();

System.***out***.print("Enter new artist (leave blank for no change): ");

artist = scanner.nextLine();

System.***out***.print("Enter new year (leave blank for no change): ");

String yearInput = scanner.nextLine();

year = (yearInput.isEmpty()) ? 0 : Integer.*parseInt*(yearInput);

System.***out***.print("Enter new description (leave blank for no change): ");

description = scanner.nextLine();

gallery.updateArtwork(artworkId, title, artist, year, description);

break;

case "3":

System.***out***.print("Enter artwork ID to delete: ");

artworkId = scanner.nextLine();

gallery.deleteArtwork(artworkId);

break;

case "4":

System.***out***.print("Enter title to search (leave blank for no title filter): ");

title = scanner.nextLine();

System.***out***.print("Enter artist to search (leave blank for no artist filter): ");

artist = scanner.nextLine();

List<Artwork> results = gallery.searchArtwork(title, artist);

if (results.isEmpty()) {

System.***out***.println("No matching artworks found.");

} else {

for (Artwork art : results) {

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

}

}

break;

case "5":

System.***out***.print("Enter artwork ID to add to favorites: ");

artworkId = scanner.nextLine();

gallery.addToFavorites(artworkId);

break;

case "6":

System.***out***.print("Enter artwork ID to remove from favorites: ");

artworkId = scanner.nextLine();

gallery.removeFromFavorites(artworkId);

break;

case "7":

System.***out***.println("\nGallery:");

gallery.viewGallery();

break;

case "8":

System.***out***.println("\nFavorites:");

gallery.viewFavorites();

break;

case "9":

System.***out***.println("Exiting program.");

scanner.close();

return;

default:

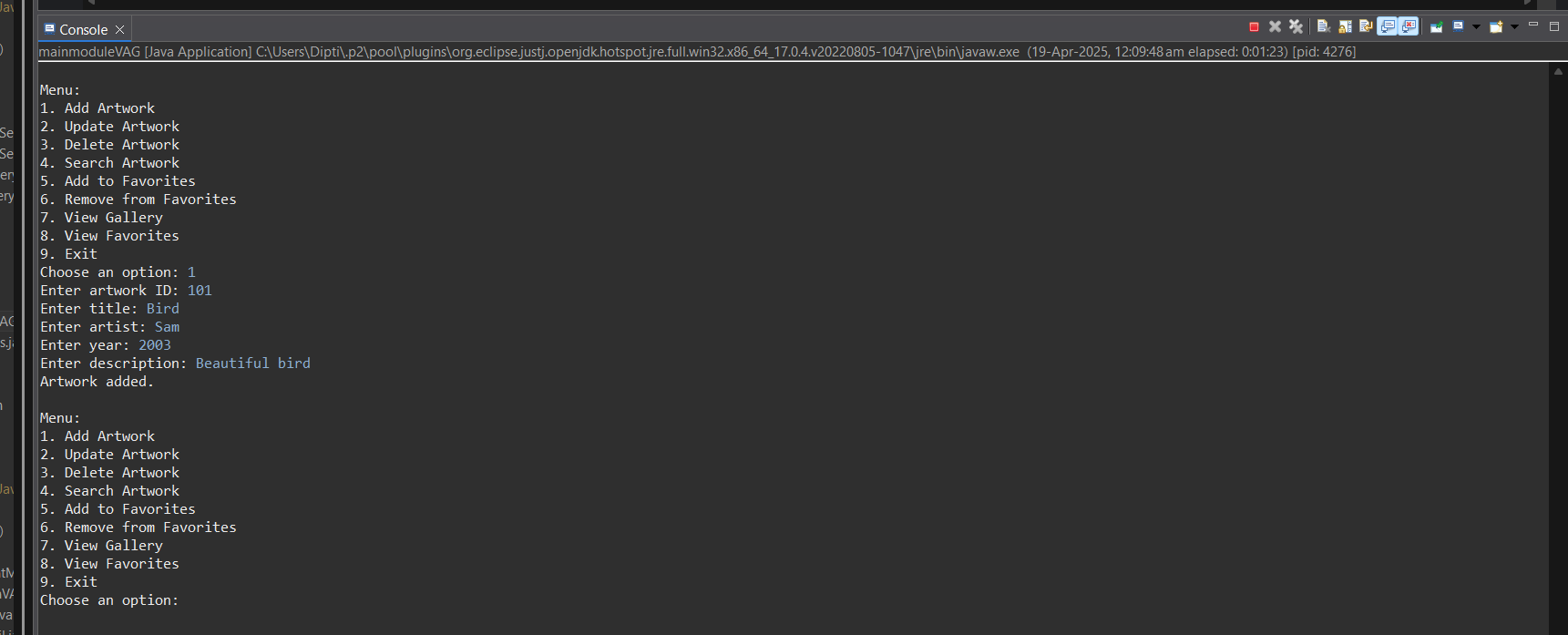
System.***out***.println("Invalid choice, please try again.");

}

}

}

}



10. Unit Testing

Creating Unit 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. 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.

package VAGJunit;

import static org.junit.jupiter.api.Assertions.\*;

import java.time.LocalDate;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;

import daoVAG.VirtualArtGalleryImpl;

import VAG.Artwork;

*@TestMethodOrder*(OrderAnnotation.class)

public class ArtworkServiceTest {

private static VirtualArtGalleryImpl *service*;

*@BeforeAll*

public static void setup() {

*service* = new VirtualArtGalleryImpl();

}

*@Test*

*@Order*(1)

public void testAddArtwork() {

Artwork artwork = new Artwork(101, "Starry Night", "Night sky over village",

LocalDate.*of*(1889, 6, 1), "Oil on canvas", "url\_to\_image", 1);

*assertTrue*(*service*.addArtwork(artwork));

}

*@Test*

*@Order*(2)

public void testUpdateArtwork() {

Artwork artwork = new Artwork(101, "Starry Night Edited", "Updated description",

LocalDate.*of*(1889, 6, 1), "Oil on canvas", "url\_to\_new\_image", 1);

*assertTrue*(*service*.updateArtwork(artwork));

}

*@Test*

*@Order*(3)

public void testGetArtworkById() {

Artwork artwork = *service*.getArtworkById(101);

*assertNotNull*(artwork);

}

*@Test*

*@Order*(4)

public void testSearchArtworks() {

*assertFalse*(*service*.searchArtworks("Night").isEmpty());

}

*@Test*

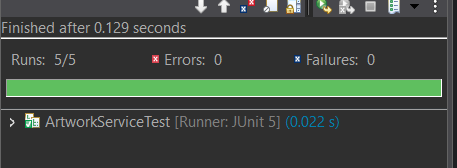
*@Order*(5)

public void testRemoveArtwork() {

*assertTrue*(*service*.removeArtwork(101));

}

}



2. 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.

package VAGJunit;

import static org.junit.jupiter.api.Assertions.\*;

import daoVAG.VirtualArtGalleryImpl;

import VAG.Gallery;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;

import java.util.List;

@TestMethodOrder(OrderAnnotation.class)

public class GalleryServiceTest {

private static VirtualArtGalleryImpl service;

@BeforeAll

public static void init() {

service = new VirtualArtGalleryImpl();

}

@Test

@Order(1)

public void testAddGallery() {

Gallery gallery = new Gallery(6, "Impressionist Dreams", "A collection of Impressionist art", "Paris", 2, "10:00 AM - 5:00 PM");

boolean result = service.addGallery(gallery);

assertTrue(result, "Gallery should be added successfully.");

}

@Test

@Order(2)

public void testUpdateGallery() {

Gallery updatedGallery = new Gallery(6, "Impressionist Dreams - Updated", "Updated description", "Paris", 2, "9:00 AM - 6:00 PM");

boolean result = service.updateGallery(updatedGallery);

assertTrue(result, "Gallery details should be updated.");

}

@Test

@Order(4)

public void testRemoveGallery() {

boolean result = service.removeGallery(6);

assertTrue(result, "Gallery should be removed.");

}

@Test

@Order(3)

public void testSearchGalleries() {

List<Gallery> results = service.searchGalleries("Impressionist");

assertNotNull(results, "Search should return a list (possibly empty).");

assertFalse(results.isEmpty(), "Search should return at least one gallery.");

}

}

