Case Study - Virtual Art Gallery

Package : Entity

Class : Artist

```
package entity;

import java.util.Date;

public class Artist {
   private int artistID;
   private String name;
   private String biography;
   private Date birthDate;
   private String nationality;
   private String website;
   private String contactInformation;
        public Artist() {
                super();

        }
        public Artist(int artistID, String name, String biography, Date birthDate, String
nationality, String website,
                        String contactInformation) {
                super();
                this.artistID = artistID;
                this.name = name;
                this.biography = biography;
                this.birthDate = birthDate;
                this.nationality = nationality;
                this.website = website;
                this.contactInformation = contactInformation;
        }
        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 Date getBirthDate() {
                return birthDate;
        }
        public void setBirthDate(Date 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 getContactInformation() {
                return contactInformation;
        }
        public void setContactInformation(String contactInformation) {
                this.contactInformation = contactInformation;
        }



}
```

Class : Artwork

```
package entity;

import java.util.Date;

public class Artwork {
        private int artworkID;
   private String title;
   private String description;
   private Date creationDate;
```

```
    private String medium;
    private String imageURL;
    private int artistID;
        public Artwork() {
                super();

        }
        public Artwork(int artworkID, String title, String description, Date creationDate,
String medium, String imageURL,
                        int artistID) {
                super();
                this.artworkID = artworkID;
                this.title = title;
                this.description = description;
                this.creationDate = creationDate;
                this.medium = medium;
                this.imageURL = imageURL;
                this.artistID = artistID;
        }
        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 Date getCreationDate() {
                return creationDate;
        }
        public void setCreationDate(Date 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;
	}









}
```

Class : Gallery

```
package entity;

public class Gallery {
   private int galleryID;
   private String name;
   private String description;
   private String location;
   private int curator;
   private String openingHours;
	public Gallery() {
		super();

	}
	public Gallery(int galleryID, String name, String description, String location, int
curator, String openingHours) {
		super();
		this.galleryID = galleryID;
		this.name = name;
		this.description = description;
		this.location = location;
		this.curator = curator;
		this.openingHours = openingHours;
```

```
	}
	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 getCurator() {
		return curator;
	}
	public void setCurator(int curator) {
		this.curator = curator;
	}
	public String getOpeningHours() {
		return openingHours;
	}
	public void setOpeningHours(String openingHours) {
		this.openingHours = openingHours;
	}



}
```

Class : User

```
package entity;
```

```
import java.util.Date;
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 Date dateOfBirth;
   private String profilePicture;
   private List<Integer> favoriteArtworks;
        public User() {
                super();

        }
        public User(int userID, String username, String password, String email, String
firstName, String lastName,
                        Date dateOfBirth, String profilePicture, List<Integer>
favoriteArtworks) {
                super();
                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;
        }
        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 Date getDateOfBirth() {
		return dateOfBirth;
	}
	public void setDateOfBirth(Date 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;
	}



}
```

Package : dao

Interface : Gallerydao

```
package dao;

import java.util.List;

import entity.Gallery;

public interface GalleryDAO {
   boolean addGallery(Gallery gallery);
   boolean updateGallery(Gallery gallery);
   boolean removeGallery(int galleryId);
   List<Gallery> searchGalleries(String keyword);
   Gallery getGalleryById(int galleryId); // returns null if not found
}
```

Interface : virtualartgallary

```
package dao;

import exception.ArtWorkNotFoundException;
import exception.UserNotFoundException;

import java.util.List;

import entity.Artwork;

public interface VirtualArtGallery {
   boolean addArtwork(Artwork artwork);
   boolean updateArtwork(Artwork artwork) throws ArtWorkNotFoundException;
   boolean removeArtwork(int artworkID) throws ArtWorkNotFoundException;
   Artwork getArtworkById(int artworkID) throws ArtWorkNotFoundException;
   List<Artwork> searchArtworks(String keyword);

   boolean addArtworkToFavorite(int userId, int artworkId)
      throws UserNotFoundException, ArtWorkNotFoundException;
   boolean removeArtworkFromFavorite(int userId, int artworkId)
      throws UserNotFoundException, ArtWorkNotFoundException;
   List<Artwork> getUserFavoriteArtworks(int userId) throws UserNotFoundException;
}
```

Class : gallarydaoimpl

```
package dao;

import exception.DbConnectionException;
import util.DbConnectionUtil;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import entity.Gallery;

public class GalleryDAOImpl implements GalleryDAO {
   private Connection conn;

   public GalleryDAOImpl() throws DbConnectionException {
      conn = DbConnectionUtil.getDbConnection();
   }

   @Override
   public boolean addGallery(Gallery g) {
      String sql = "INSERT INTO Gallery (Name, Description, Location, CuratorID,
OpeningHours) VALUES (?, ?, ?, ?, ?)";
      try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
         pstmt.setString(1, g.getName());
         pstmt.setString(2, g.getDescription());
         pstmt.setString(3, g.getLocation());
         pstmt.setInt(4, g.getCurator());
         pstmt.setString(5, g.getOpeningHours());
         return pstmt.executeUpdate() > 0;
      } catch (SQLException e) {
         e.printStackTrace(); return false;
      }
   }

   @Override
   public boolean updateGallery(Gallery g) {
      String sql = "UPDATE Gallery SET Name = ?, Description = ?, Location = ?, CuratorID
= ?, OpeningHours = ? WHERE GalleryID = ?";
      try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
         pstmt.setString(1, g.getName());
         pstmt.setString(2, g.getDescription());
         pstmt.setString(3, g.getLocation());
         pstmt.setInt(4, g.getCurator());
         pstmt.setString(5, g.getOpeningHours());
```

```
            pstmt.setInt(6, g.getGalleryID());
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace(); return false;
        }
    }

    @Override
    public boolean removeGallery(int galleryId) {
        String sql = "DELETE FROM Gallery WHERE GalleryID = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, galleryId);
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace(); return false;
        }
    }

    @Override
    public List<Gallery> searchGalleries(String keyword) {
        List<Gallery> list = new ArrayList<>();
        String sql = "SELECT * FROM Gallery WHERE Name LIKE ? OR Description LIKE ? OR
Location LIKE ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            String pattern = "%" + keyword + "%";
            for (int i = 1; i <= 3; i++) pstmt.setString(i, pattern);
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                list.add(new Gallery(
                        rs.getInt("GalleryID"),
                        rs.getString("Name"),
                        rs.getString("Description"),
                        rs.getString("Location"),
                        rs.getInt("CuratorID"),
                        rs.getString("OpeningHours")
                ));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    @Override
    public Gallery getGalleryById(int galleryId) {
        String sql = "SELECT * FROM Gallery WHERE GalleryID = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
```

```
            pstmt.setInt(1, galleryId);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return new Gallery(
                        rs.getInt("GalleryID"),
                        rs.getString("Name"),
                        rs.getString("Description"),
                        rs.getString("Location"),
                        rs.getInt("CuratorID"),
                        rs.getString("OpeningHours")
                );
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Class : virtualartgallaryimpl

```
package dao;

import exception.ArtWorkNotFoundException;
import exception.DbConnectionException;
import exception.UserNotFoundException;
import util.DbConnectionUtil;
import util.HexaConstants;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

import entity.Artwork;

public class VirtualArtGalleryImpl implements VirtualArtGallery {
    private Connection connection;

    public VirtualArtGalleryImpl() throws DbConnectionException {
        connection = DbConnectionUtil.getDbConnection();
    }

    @Override
    public boolean addArtwork(Artwork artwork) {
        String sql = HexaConstants.ADD_ARTWORK_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            setArtworkParameters(pstmt, artwork);
```

```
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            handleSQLException(e);
            return false;
        }
    }

    @Override
    public boolean updateArtwork(Artwork artwork) throws ArtWorkNotFoundException {
        String sql = HexaConstants.UPDATE_ARTWORK_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            setArtworkParameters(pstmt, artwork);
            pstmt.setInt(7, artwork.getArtworkID());

            int affectedRows = pstmt.executeUpdate();
            if (affectedRows == 0) {
                throw new ArtWorkNotFoundException("Artwork with ID " +
artwork.getArtworkID() + " not found");
            }
            return true;
        } catch (SQLException e) {
            handleSQLException(e);
            return false;
        }
    }

    @Override
    public boolean removeArtwork(int artworkID) throws ArtWorkNotFoundException {
        String sql = HexaConstants.DELETE_ARTWORK_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, artworkID);
            int affectedRows = pstmt.executeUpdate();
            if (affectedRows == 0) {
                throw new ArtWorkNotFoundException("Artwork with ID " + artworkID + " not
found");
            }
            return true;
        } catch (SQLException e) {
            handleSQLException(e);
            return false;
        }
    }

    @Override
    public Artwork getArtworkById(int artworkID) throws ArtWorkNotFoundException {
        String sql = HexaConstants.GET_ARTWORK_BY_ID_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
```

```
            pstmt.setInt(1, artworkID);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                return createArtworkFromResultSet(rs);
            } else {
                throw new ArtWorkNotFoundException("Artwork with ID " + artworkID + " not
found");
            }
        } catch (SQLException e) {
            throw new ArtWorkNotFoundException("Error retrieving artwork: " +
e.getMessage());
        }
    }

    @Override
    public List<Artwork> searchArtworks(String keyword) {
        List<Artwork> results = new ArrayList<>();
        String sql = HexaConstants.SEARCH_ARTWORK_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            String searchPattern = "%" + keyword + "%";
            for (int i = 1; i <= 3; i++) {
                pstmt.setString(i, searchPattern);
            }

            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                results.add(createArtworkFromResultSet(rs));
            }
        } catch (SQLException e) {
            handleSQLException(e);
        }
        return results;
    }

    @Override
    public boolean addArtworkToFavorite(int userId, int artworkId)
            throws UserNotFoundException, ArtWorkNotFoundException {
        validateUserAndArtwork(userId, artworkId);

        String sql = HexaConstants.ADD_FAVORITE_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, userId);
            pstmt.setInt(2, artworkId);
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            handleSQLException(e);
```

```
            return false;
        }
    }

    @Override
    public boolean removeArtworkFromFavorite(int userId, int artworkId)
            throws UserNotFoundException, ArtWorkNotFoundException {
        validateUserAndArtwork(userId, artworkId);

        String sql = HexaConstants.REMOVE_FAVORITE_QRY;
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, userId);
            pstmt.setInt(2, artworkId);
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            handleSQLException(e);
            return false;
        }
    }

    @Override
    public List<Artwork> getUserFavoriteArtworks(int userId) throws
UserNotFoundException {
        if (!isUserExists(userId)) {
            throw new UserNotFoundException("User with ID " + userId + " not found");
        }

        List<Artwork> favorites = new ArrayList<>();
        String sql = HexaConstants.GET_USER_FAVORITES_QRY;

        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, userId);
            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                favorites.add(createArtworkFromResultSet(rs));
            }
        } catch (SQLException e) {
            handleSQLException(e);
        }
        return favorites;
    }

    // ============================
    // ===== Helper Methods ======
    // ============================
```

```
    private Artwork createArtworkFromResultSet(ResultSet rs) throws SQLException {
        return new Artwork(
                rs.getInt(HexaConstants.COL_ARTWORK_ID),
                rs.getString(HexaConstants.COL_TITLE),
                rs.getString(HexaConstants.COL_DESCRIPTION),
                rs.getDate(HexaConstants.COL_CREATION_DATE),
                rs.getString(HexaConstants.COL_MEDIUM),
                rs.getString(HexaConstants.COL_IMAGE_URL),
                rs.getInt(HexaConstants.COL_ARTIST_ID)
        );
    }

    private void setArtworkParameters(PreparedStatement pstmt, Artwork artwork) throws
SQLException {
        pstmt.setString(1, artwork.getTitle());
        pstmt.setString(2, artwork.getDescription());
        pstmt.setDate(3, new java.sql.Date(artwork.getCreationDate().getTime()));
        pstmt.setString(4, artwork.getMedium());
        pstmt.setString(5, artwork.getImageURL());
        pstmt.setInt(6, artwork.getArtistID());
    }

    private void validateUserAndArtwork(int userId, int artworkId)
            throws UserNotFoundException, ArtWorkNotFoundException {
        if (!isUserExists(userId)) {
            throw new UserNotFoundException("User with ID " + userId + " not found");
        }
        if (!isArtworkExists(artworkId)) {
            throw new ArtWorkNotFoundException("Artwork with ID " + artworkId + " not
found");
        }
    }

    private boolean isUserExists(int userId) {
        return checkEntityExists("User", HexaConstants.COL_USER_ID, userId);
    }

    private boolean isArtworkExists(int artworkId) {
        return checkEntityExists("Artwork", HexaConstants.COL_ARTWORK_ID, artworkId);
    }

    private boolean checkEntityExists(String table, String column, int id) {
        String sql = String.format(HexaConstants.CHECK_ENTITY_EXISTS_TEMPLATE, column,
table, column);
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            return pstmt.executeQuery().next();
```

```
        } catch (SQLException e) {
            handleSQLException(e);
            return false;
        }
    }

    private void handleSQLException(SQLException e) {
        System.err.println("SQL Error: " + e.getMessage());
        System.err.println("SQL State: " + e.getSQLState());
        System.err.println("Error Code: " + e.getErrorCode());
    }
}
```

Package : util

Class : DBConnection

```
package util;

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

import exception.DbConnectionException;

public class DbConnectionUtil {

    static {
        try {
            Class.forName(DbProperties.getDriver());
        } catch (ClassNotFoundException e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }

    public static Connection getDbConnection() throws DbConnectionException {
        Connection con = null;
        try {
            con =
DriverManager.getConnection(DbProperties.getDbUrl(),DbProperties.getProps());
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
            throw new DbConnectionException(HexaConstants.CANNOT_OPEN_CONNECTION);
        }
        return con;
```

```
    }

    public static void closeConnection(Connection con) {
        try {
            if(con !=null)  con.close();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }

    }
}
```

Class : Dbproperty

```
package util;

import java.io.IOException;
import java.util.Properties;
import java.util.Scanner;

public class DbProperties {
    private static Properties props = null;
    public Scanner scanner = new Scanner(System.in);
    static {
        props = getDbProperties();
    }
   private static Properties getDbProperties() {
        Properties props = new Properties();
        try {

props.load(DbProperties.class.getClassLoader().getResourceAsStream(HexaConstants.DB_
FILE_NAME));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
        return props;
    }

 public static String getDriver() {
        return props.getProperty(HexaConstants.DB_DRIVER);

    }
    public static String getDbUrl() {
        return props.getProperty(HexaConstants.DB_URL);
```

```
    }
    public static Properties getProps() {
        return props;

    }
}
```

Package : main

Class : mainmeathod

```
package main;

import dao.VirtualArtGallery;
import dao.VirtualArtGalleryImpl;
import entity.Artwork;
import exception.ArtWorkNotFoundException;
import exception.DbConnectionException;
import exception.UserNotFoundException;
import util.HexaConstants;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

public class MainModule {
    private static final VirtualArtGallery galleryService;

    static {
        try {
            galleryService = new VirtualArtGalleryImpl();
        } catch (DbConnectionException e) {
            throw new RuntimeException(e);
        }
    }

    private static final Scanner scanner = new Scanner(System.in);
    private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd");

    public static void main(String[] args) {
        while (true) {
            try {
                displayMenu();
                int choice = getIntInput(HexaConstants.PROMPT_CHOICE);
```

```
            switch (choice) {
                case 1 -> addArtwork();
                case 2 -> updateArtwork();
                case 3 -> removeArtwork();
                case 4 -> searchArtworkById();
                case 5 -> searchArtworksByKeyword();
                case 6 -> addToFavorites();
                case 7 -> removeFromFavorites();
                case 8 -> viewFavorites();
                case 9 -> {
                    System.out.println(HexaConstants.EXIT_MSG);
                    System.exit(0);
                }
                default -> System.out.println(HexaConstants.INVALID_CHOICE_MSG);
            }
        } catch (ArtWorkNotFoundException | UserNotFoundException e) {
            System.out.println(HexaConstants.USER_ERROR_PREFIX + e.getMessage());
        } catch (Exception e) {
            System.out.println(HexaConstants.UNEXPECTED_ERROR_PREFIX +
e.getMessage());
        }
    }
}

private static void displayMenu() {
    System.out.println(HexaConstants.HEADER_MAIN_MENU);
    System.out.println(HexaConstants.MENU_OPTION_1);
    System.out.println(HexaConstants.MENU_OPTION_2);
    System.out.println(HexaConstants.MENU_OPTION_3);
    System.out.println(HexaConstants.MENU_OPTION_4);
    System.out.println(HexaConstants.MENU_OPTION_5);
    System.out.println(HexaConstants.MENU_OPTION_6);
    System.out.println(HexaConstants.MENU_OPTION_7);
    System.out.println(HexaConstants.MENU_OPTION_8);
    System.out.println(HexaConstants.MENU_OPTION_9);
}

private static int getIntInput(String prompt) {
    while (true) {
        try {
            System.out.print(prompt);
            return Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println(HexaConstants.INVALID_NUMBER_INPUT);
        }
    }
}
```

```
    private static Date getDateInput(String prompt) {
        while (true) {
            try {
                System.out.print(prompt + " (yyyy-mm-dd): ");
                return dateFormat.parse(scanner.nextLine());
            } catch (ParseException e) {
                System.out.println(HexaConstants.INVALID_DATE_INPUT);
            }
        }
    }

    private static void addArtwork() {
        System.out.println(HexaConstants.HEADER_ADD_ARTWORK);
        int id = getIntInput(HexaConstants.PROMPT_ARTWORK_ID);
        String title = getStringInput(HexaConstants.PROMPT_TITLE);
        String description = getStringInput(HexaConstants.PROMPT_DESCRIPTION);
        Date creationDate = getDateInput(HexaConstants.PROMPT_CREATION_DATE);
        String medium = getStringInput(HexaConstants.PROMPT_MEDIUM);
        String imageURL = getStringInput(HexaConstants.PROMPT_IMAGE_URL);
        int artistId = getIntInput(HexaConstants.PROMPT_ARTIST_ID);

        Artwork artwork = new Artwork(id, title, description, creationDate, medium,
imageURL, artistId);
        if (galleryService.addArtwork(artwork)) {
            System.out.println(HexaConstants.ADD_SUCCESS);
        } else {
            System.out.println(HexaConstants.ADD_FAILURE);
        }
    }

    private static void updateArtwork() throws ArtWorkNotFoundException {
        System.out.println(HexaConstants.HEADER_UPDATE_ARTWORK);
        int id = getIntInput(HexaConstants.UPDATE_ARTWORK_ID);
        Artwork existing = galleryService.getArtworkById(id);
        System.out.println(HexaConstants.CURRENT_DETAILS);
        displayArtwork(existing);

        String title = getStringInput(HexaConstants.PROMPT_NEW_TITLE);
        String description = getStringInput(HexaConstants.PROMPT_NEW_DESCRIPTION);
        Date creationDate =
getOptionalDate(HexaConstants.PROMPT_NEW_CREATION_DATE);
        String medium = getStringInput(HexaConstants.PROMPT_NEW_MEDIUM);
        String imageURL = getStringInput(HexaConstants.PROMPT_NEW_IMAGE_URL);
        Integer artistId = getOptionalInt(HexaConstants.PROMPT_NEW_ARTIST_ID);

        Artwork updated = new Artwork(
```

```
            id,
            title.isEmpty() ? existing.getTitle() : title,
            description.isEmpty() ? existing.getDescription() : description,
            creationDate != null ? creationDate : existing.getCreationDate(),
            medium.isEmpty() ? existing.getMedium() : medium,
            imageURL.isEmpty() ? existing.getImageURL() : imageURL,
            artistId != null ? artistId : existing.getArtistID()
        );

        if (galleryService.updateArtwork(updated)) {
            System.out.println(HexaConstants.UPDATE_SUCCESS);
        } else {
            System.out.println(HexaConstants.UPDATE_FAILURE);
        }
    }

    private static void removeArtwork() throws ArtWorkNotFoundException {
        System.out.println(HexaConstants.HEADER_REMOVE_ARTWORK);
        int artworkId = getIntInput(HexaConstants.PROMPT_ARTWORK_ID);
        if (galleryService.removeArtwork(artworkId)) {
            System.out.println(HexaConstants.REMOVE_SUCCESS);
        } else {
            System.out.println(HexaConstants.REMOVE_FAILURE);
        }
    }

    private static void searchArtworkById() throws ArtWorkNotFoundException {
        System.out.println(HexaConstants.HEADER_SEARCH_BY_ID);
        int id = getIntInput(HexaConstants.PROMPT_ARTWORK_ID);
        Artwork artwork = galleryService.getArtworkById(id);
        displayArtwork(artwork);
    }

    private static void searchArtworksByKeyword() {
        System.out.println(HexaConstants.HEADER_SEARCH_BY_KEYWORD);
        String keyword = getStringInput(HexaConstants.PROMPT_SEARCH_KEYWORD);
        List<Artwork> results = galleryService.searchArtworks(keyword);
        if (results.isEmpty()) {
            System.out.println(HexaConstants.NO_RESULTS_FOUND);
        } else {
            System.out.println("Found " + results.size() + " artworks:");
            results.forEach(MainModule::displayArtwork);
        }
    }

    private static void addToFavorites() throws UserNotFoundException,
ArtWorkNotFoundException {
```

```
        System.out.println(HexaConstants.HEADER_ADD_TO_FAV);
        int userId = getIntInput(HexaConstants.PROMPT_USER_ID);
        int artworkId = getIntInput(HexaConstants.PROMPT_ARTWORK_ID);
        if (galleryService.addArtworkToFavorite(userId, artworkId)) {
            System.out.println(HexaConstants.FAV_ADD_SUCCESS);
        } else {
            System.out.println(HexaConstants.FAV_ADD_FAILURE);
        }
    }

    private static void removeFromFavorites() throws UserNotFoundException,
ArtWorkNotFoundException {
        System.out.println(HexaConstants.HEADER_REMOVE_FROM_FAV);
        int userId = getIntInput(HexaConstants.PROMPT_USER_ID);
        int artworkId = getIntInput(HexaConstants.PROMPT_ARTWORK_ID);
        if (galleryService.removeArtworkFromFavorite(userId, artworkId)) {
            System.out.println(HexaConstants.FAV_REMOVE_SUCCESS);
        } else {
            System.out.println(HexaConstants.FAV_REMOVE_FAILURE);
        }
    }

    private static void viewFavorites() throws UserNotFoundException {
        System.out.println(HexaConstants.HEADER_VIEW_FAV);
        int userId = getIntInput(HexaConstants.PROMPT_USER_ID);
        List<Artwork> favorites = galleryService.getUserFavoriteArtworks(userId);
        if (favorites.isEmpty()) {
            System.out.println(HexaConstants.NO_FAVORITES);
        } else {
            System.out.println(HexaConstants.FAVORITE_LIST_HEADER);
            favorites.forEach(MainModule::displayArtwork);
        }
    }

    private static String getStringInput(String prompt) {
        System.out.print(prompt);
        return scanner.nextLine();
    }

    private static Date getOptionalDate(String prompt) {
        System.out.print(prompt + " (press enter to skip): ");
        String input = scanner.nextLine();
        if (input.isEmpty()) return null;
        try {
            return dateFormat.parse(input);
        } catch (ParseException e) {
            System.out.println(HexaConstants.INVALID_DATE_KEEP_CURRENT);
```

```
            return null;
        }
    }

    private static Integer getOptionalInt(String prompt) {
        System.out.print(prompt + " (press enter to skip): ");
        String input = scanner.nextLine();
        if (input.isEmpty()) return null;
        try {
            return Integer.parseInt(input);
        } catch (NumberFormatException e) {
            System.out.println(HexaConstants.INVALID_NUMBER_KEEP_CURRENT);
            return null;
        }
    }

    private static void displayArtwork(Artwork artwork) {
        System.out.println("\n=== Artwork Details ===");
        System.out.println("ID: " + artwork.getArtworkID());
        System.out.println("Title: " + artwork.getTitle());
        System.out.println("Description: " + artwork.getDescription());
        System.out.println("Creation Date: " +
dateFormat.format(artwork.getCreationDate()));
        System.out.println("Medium: " + artwork.getMedium());
        System.out.println("Image URL: " + artwork.getImageURL());
        System.out.println("Artist ID: " + artwork.getArtistID());
    }
}
```

Package : Test

Class : Artwork Service Test

```
package com.hexaware.test;

import dao.VirtualArtGallery;
import dao.VirtualArtGalleryImpl;
import entity.Artwork;
import exception.ArtWorkNotFoundException;
import exception.DbConnectionException;

import org.junit.jupiter.api.*;

import java.util.Date;
import java.util.List;

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

```
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class ArtworkServiceTest {

   private static VirtualArtGallery galleryService;
   private static int generatedArtworkId; // to store actual DB ID

   @BeforeAll
   public static void init() throws DbConnectionException {
      galleryService = new VirtualArtGalleryImpl();
   }

   @Test
   @Order(1)
   public void testAddArtwork_success() {
      Artwork artwork = new Artwork(0, "JUnit Test Title", "JUnit Test Description", new
Date(),
            "Oil", "junit_test.jpg", 1);
      boolean result = galleryService.addArtwork(artwork);
      assertTrue(result, "Artwork should be added successfully");


      List<Artwork> found = galleryService.searchArtworks("JUnit Test Title");
      assertFalse(found.isEmpty(), "Inserted artwork should be found in search");
      generatedArtworkId = found.get(0).getArtworkID();
   }

   @Test
   @Order(2)
   public void testUpdateArtwork_success() throws ArtWorkNotFoundException {
      Artwork updated = new Artwork(generatedArtworkId, "Updated Title", "Updated
Description", new Date(),
            "Acrylic", "updated_url.jpg", 1);
      boolean result = galleryService.updateArtwork(updated);
      assertTrue(result, "Artwork should be updated successfully");

      Artwork fetched = galleryService.getArtworkById(generatedArtworkId);
      assertEquals("Updated Title", fetched.getTitle());
   }

   @Test
   @Order(3)
   public void testGetArtworkById_success() throws ArtWorkNotFoundException {
      Artwork artwork = galleryService.getArtworkById(generatedArtworkId);
      assertNotNull(artwork);
      assertEquals(generatedArtworkId, artwork.getArtworkID());
   }
```

```
    @Test
    @Order(4)
    public void testSearchArtworks_found() {
        List<Artwork> results = galleryService.searchArtworks("Updated");
        assertFalse(results.isEmpty(), "Search should return results for keyword 'Updated'");
    }

    @Test
    @Order(5)
    public void testRemoveArtwork_success() throws ArtWorkNotFoundException {
        boolean result = galleryService.removeArtwork(generatedArtworkId);
        assertTrue(result, "Artwork should be removed successfully");
    }

    @Test
    @Order(6)
    public void testGetArtworkById_notFound() {
        assertThrows(ArtWorkNotFoundException.class, () -> {
            galleryService.getArtworkById(generatedArtworkId);
        });
    }
}
```

Class : Gallary Service Test

```
package com.hexaware.test;

import dao.GalleryDAO;
import dao.GalleryDAOImpl;
import entity.Gallery;
import exception.DbConnectionException;

import org.junit.jupiter.api.*;

import java.util.List;

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

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class GalleryServiceTest {

    private static GalleryDAO galleryDAO;
    private static int generatedGalleryId;

    @BeforeAll
    public static void init() throws DbConnectionException {
```

```
        galleryDAO = new GalleryDAOImpl();
    }

    @Test
    @Order(1)
    public void testAddGallery_success() {
        Gallery gallery = new Gallery(0, "JUnit Test Gallery", "JUnit Description", "Mumbai",
                1, "9am-6pm");

        boolean result = galleryDAO.addGallery(gallery);
        assertTrue(result, "Gallery should be added successfully");

        // Fetch back to confirm and get generated ID
        List<Gallery> results = galleryDAO.searchGalleries("JUnit");
        assertFalse(results.isEmpty(), "Gallery should be found");
        generatedGalleryId = results.get(0).getGalleryID();
    }

    @Test
    @Order(2)
    public void testUpdateGallery_success() {
        Gallery updated = new Gallery(generatedGalleryId, "Updated Gallery Name",
"Updated Desc",
                "Pune", 1, "10am-5pm");

        boolean result = galleryDAO.updateGallery(updated);
        assertTrue(result, "Gallery should be updated successfully");

        Gallery fetched = galleryDAO.getGalleryById(generatedGalleryId);
        assertEquals("Updated Gallery Name", fetched.getName());
    }

    @Test
    @Order(3)
    public void testSearchGallery_found() {
        List<Gallery> results = galleryDAO.searchGalleries("Updated");
        assertFalse(results.isEmpty(), "Search should return at least one result");
    }

    @Test
    @Order(4)
    public void testRemoveGallery_success() {
        boolean result = galleryDAO.removeGallery(generatedGalleryId);
        assertTrue(result, "Gallery should be removed successfully");
    }

    @Test
```

```
    @Order(5)
    public void testGetGalleryById_notFound() {
        Gallery gallery = galleryDAO.getGalleryById(generatedGalleryId);
        assertNull(gallery, "After removal, gallery should not be found");
    }
}
```

Outputs :

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 1
```

1.

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 1

=== Add New Artwork ===
Artwork ID: 1
Title: Sunset
Description: Sunset In Mountains
Creation Date (yyyy-mm-dd):  (yyyy-mm-dd): 2025-02-04
Medium: Canvas
Image URL: www.img.com
Artist ID: 2
Artwork added successfully!
```

2.

```
=== Artwork Details ===
ID: 14
Title: Sunset
Description: Sunset In Mountains
Creation Date: 2025-02-04
Medium: Canvas
Image URL: www.img.com
Artist ID: 2
New Title (leave blank to keep current): Sunset In Mountains
New Description (leave blank to keep current):
New Creation Date (press enter to skip):
New Medium (leave blank to keep current):
New Image URL (leave blank to keep current):
New Artist ID (press enter to skip):
Artwork updated successfully!
```

3.

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 3

=== Remove Artwork ===
Artwork ID: 2
Artwork removed successfully!
```

4.

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 4

=== Search Artwork by ID ===
Artwork ID: 4

=== Artwork Details ===
ID: 4
Title: Guernica
Description: Depicts the bombing of Guernica during Spanish Civil War
Creation Date: 1937-04-26
Medium: Oil on canvas
Image URL: guernica.jpg
Artist ID: 3
```

5.

```
=== Search Artworks ===
Enter search keyword: sunset
Found 2 artworks:

=== Artwork Details ===
ID: 7
Title: sunset
Description: sunset
Creation Date: 2025-12-04
Medium: Canvas
Image URL: www.img.com
Artist ID: 3

=== Artwork Details ===
ID: 14
Title: Sunset In Mountains
Description: Sunset In Mountains
Creation Date: 2025-02-04
Medium: Canvas
Image URL: www.img.com
Artist ID: 2
```

6.

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 6

=== Add to Favorites ===
User ID: 2
Artwork ID: 3
Artwork added to favorites!
```

7.

```
=== Virtual Art Gallery Management Syste
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 7

=== Remove from Favorites ===
User ID: 2
Artwork ID: 3
Artwork removed from favorites!
```

8.

```
=== Virtual Art Gallery Management System ===
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Search Artwork by ID
5. Search Artworks by Keyword
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorites
9. Exit
Enter your choice: 8

=== View Favorites ===
User ID: 2
Favorite artworks:

=== Artwork Details ===
ID: 3
Title: Starry Night
Description: A famous night landscape by Van Gogh
Creation Date: 1889-06-01
Medium: Oil on canvas
Image URL: starry.jpg
Artist ID: 2
```