

Case Study: Virtual Art Gallery

By -- Dhanush A

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.

```
create database artgal;

use artgal;

> create table artwork(
    ArtworkID INT primary key auto_increment,
    ArtistID int,
    Title VARCHAR(45),
    Description VARCHAR(255),
    CreationDate date,
    Medium VARCHAR(25),
    ImageURL VARCHAR(255));

> create table artist(
    ArtistID INT primary key auto_increment,
    Name VARCHAR(45),
    Biography VARCHAR(255),
    BirthDate date,
    Nationality VARCHAR(25),
    Website VARCHAR(255),
    ContactInformation VARCHAR(255));
```

```

create table user(
    UserID INT primary key auto_increment,
    Username VARCHAR(45),
    Password VARCHAR(255),
    FirstName VARCHAR(45),
    LastName VARCHAR(45),
    DateOfBirth date,
    ProfilePicture blob,
    FavoriteArtworks VARCHAR(255));

create table gallery(
    GalleryID INT primary key auto_increment,
    Name VARCHAR(45),
    Description VARCHAR(255),
    Location VARCHAR(45),
    Curator int,
    OpeningHours VARCHAR(45));

```

```

alter table artwork add foreign key(ArtistID) references artist(ArtistID);
alter table gallery add foreign key(Curator) references artist(ArtistID);

```

```

INSERT INTO user ('UserID', 'Username', 'Password', 'FirstName', 'LastName', 'DateOfBirth', 'FavoriteArtworks') VALUES ('1', 'Dhanush', '1234', 'Dhanush', 'Ashok', '2001-05-28', '1,2,4,5');

insert into artgal.artist (ArtistID, Name, Biography, BirthDate, Nationality, Website, ContactInformation) values
(1,
"Leonardo da Vinci",
"Italian polymath of the Renaissance whose areas of interest included invention, drawing, painting, sculpture, architecture, science, music, mathematics, engineering, literature, anatomy, geol
"1986-04-15",
"Italian",
"https://leonardodavinci.net/",
"contact@leonardodavinci.net"),
(2,
"Vincent van Gogh",
"Dutch post-impressionist painter who is among the most famous and influential figures in the history of Western art. In just over a decade, he created about 2,100 artworks, including around 8
"1990-03-30",
"Dutch",
"www.vangoghmuseum.nl/",
"contact@vangoghmuseum.nl"),
(3,
"Frida Kahlo",
"Mexican painter known for her many portraits and self-portraits characterized by vibrant colors and a mix of realism and symbolism. She was associated with the Surrealist movement.",
"1985-06-06",
"Mexican",
"https://www.fridakahlo.org/",
"contact@fridakahlo.org");

```

```
INSERT INTO artgal.artwork (ArtworkID, ArtistID, Title, Description, CreationDate, Medium, ImageURL)
VALUES
(1, 1, 'Mona Lisa', 'The famous portrait of a woman with an enigmatic smile.', '2001-05-28', 'Oil on poplar panel', 'https://leonardodavin
(2, 1, 'The Last Supper', 'A depiction of the Last Supper of Jesus with his disciples.', '2002-01-04', 'Fresco', 'https://leonardodavinci.
(3, 1, 'Vitruvian Man', 'A study of the proportions of the human body based on Vitruvius\' principles.', '2001-06-24', 'Pen and ink with w
(4, 2, 'Starry Night', 'A night sky filled with swirling clouds and a bright crescent moon.', '2000-01-30', 'Oil on canvas', 'https://www.
(5, 2, 'Sunflowers', 'A series of still life paintings depicting sunflowers in various stages of bloom.', '2000-08-24', 'Oil on canvas', '
(6, 2, 'The Bedroom', 'A depiction of Van Gogh\'s bedroom in the Yellow House in Arles, France.', '2002-06-22', 'Oil on canvas', 'https://
(7, 3, 'The Two Fridas', 'A double self-portrait depicting two different aspects of Frida\'s personality.', '2001-06-06', 'Oil on canvas',
(8, 3, 'Self-Portrait with Thorn Necklace and Hummingbird', 'A self-portrait with symbolic elements, including a thorn necklace and a humm

INSERT INTO `artgal`.`gallery` (`GalleryID`, `Name`, `Description`, `Location`, `Curator`, `OpeningHours`, `Artworks`) VALUES
('1', 'Art Gallery A', 'Contemporary art gallery featuring modern masterpieces.', '123 Main Street, Cityville', '1', 'Monday to Friday: 10
INSERT INTO `artgal`.`gallery` (`GalleryID`, `Name`, `Description`, `Location`, `Curator`, `OpeningHours`, `Artworks`) VALUES
('2', 'Gallery XYZ', 'Eclectic gallery showcasing a diverse range of artistic expressions.', '456 Art Avenue, Townsville', '2', 'Tuesday t
INSERT INTO `artgal`.`gallery` (`GalleryID`, `Name`, `Description`, `Location`, `Curator`, `OpeningHours`, `Artworks`) VALUES
('3', 'Modern Art Hub', 'Cutting-edge gallery pushing the boundaries of contemporary art.', '789 Gallery Street, Artropolis', '3', 'Wednes
```

Artist Table:

| | ArtistID | Name | Biography | BirthDate | Nationality | Website | ContactInformation |
|--|----------|----------|---------------|------------|-------------|------------|----------------------|
| | 1 | Leon... | Italian po... | 1986-04... | Italian | https:/... | contact@leonardo... |
| | 2 | Vinc... | Dutch po... | 1990-03... | Dutch | www.v... | contact@vangogh... |
| | 3 | Frida... | Mexican ... | 1985-06... | Mexican | https:/... | contact@fridakahl... |

Artwork Table:

| | ArtworkID | ArtistID | Title | Description | CreationDate | Medium | ImageURL | NoOfFavorites |
|---|-----------|----------|---|---|--------------|--|---|---------------|
| ▶ | 1 | 1 | Mona Lisa | The famous portrait of a woman with an enigma... | 2001-05-20 | Oil on poplar panel | https://leonardodavind.net/images/paintings/m... | 1 |
| | 2 | 1 | The Last Supper | A depiction of the Last Supper of Jesus with his ... | 2002-01-04 | Fresco | https://leonardodavind.net/images/paintings/th... | 1 |
| | 3 | 1 | Vitruvian Man | A study of the proportions of the human body b... | 2001-08-24 | Pen and ink with wash over metalpoint on paper | https://leonardodavind.net/images/drawings/vi... | 0 |
| | 4 | 2 | Starry Night | A night sky filled with swirling clouds and a brigh... | 2000-01-30 | Oil on canvas | https://www.vangoghmuseum.nl/frida/image... | 1 |
| | 5 | 2 | Sunflowers | A series of still life paintings depicting sunflower... | 2000-08-24 | Oil on canvas | https://www.vangoghmuseum.nl/frida/image... | 1 |
| | 6 | 2 | The Bedroom | A depiction of Van Gogh's bedroom in the Yellow... | 2002-06-22 | Oil on canvas | https://www.fridakahlo.org/images/paintings/th... | 0 |
| | 7 | 3 | The Two Fridas | A double self-portrait depicting two different as... | 2001-06-06 | Oil on canvas | https://www.fridakahlo.org/images/paintings/th... | 0 |
| | 8 | 3 | Self-Portrait with Thorn Necklace and Hummingbird | A self-portrait with symbolic elements, including ... | 2001-08-28 | Oil on canvas | https://www.fridakahlo.org/images/paintings/s... | 7,8 |

User Table:

| | UserID | Username | Password | FirstName | LastName | DateOfBirth | ProfilePicture | FavoriteArtworks |
|---|--------|----------|----------|-----------|----------|-------------|----------------|------------------|
| ▶ | 1 | Dhanush | 1234 | Dhanush | ashok | 2001-05-28 | NULL | 1,2,4,5 |

Gallery Table:

| | GalleryID | Name | Description | Location | Curator | OpeningHours | Artworks |
|---|-----------|----------------|---|--------------------------------|---------|--|----------|
| ▶ | 1 | Art Gallery A | Contemporary art gallery featuring modern mas... | 123 Main Street, Cityville | 1 | Monday to Friday: 10 AM - 6 PM, Saturday: 12 ... | 1,2,3 |
| | 2 | Gallery XYZ | Eclectic gallery showcasing a diverse range of a... | 456 Art Avenue, Townsville | 2 | Tuesday to Saturday: 11 AM - 7 PM, Sunday: 1 ... | 4,5,6 |
| | 3 | Modern Art Hub | Cutting-edge gallery pushing the boundaries of ... | 789 Gallery Street, Artropolis | 3 | Wednesday to Sunday: 12 PM - 8 PM, Closed o... | 7,8 |

Coding

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

```
class Artwork:
    def __init__(self):
        self._artworkid = None
        self._artistid = None
        self._title = None
        self._description = None
        self._creationdate = None
        self._medium = None
        self._imageurl = None
```

3 usages (2 dynamic)

```
@property
def artworkid(self):
    return self._artworkid
```

5 usages (2 dynamic)

```
@artworkid.setter
def artworkid(self, value):
    self._artworkid = value
```

3 usages (2 dynamic)

```
@property
def artistid(self):
    return self._artistid
```

5 usages (2 dynamic)

```
@artistid.setter
def artistid(self, value):
    self._artistid = value
```

3 usages (2 dynamic)

```
@property
def title(self):
    return self._title
```

5 usages (2 dynamic)

```
@title.setter  
def title(self, value):  
    self._title = value
```

5 usages (4 dynamic)

```
@property  
def description(self):  
    return self._description
```

7 usages (4 dynamic)

```
@description.setter  
def description(self, value):  
    self._description = value
```

3 usages (2 dynamic)

```
@property  
def creationdate(self):  
    return self._creationdate
```

5 usages (2 dynamic)

```
@creationdate.setter  
def creationdate(self, value):  
    self._creationdate = value
```

3 usages (2 dynamic)

```
@property  
def medium(self):  
    return self._medium
```

5 usages (2 dynamic)

```
@medium.setter  
def medium(self, value):  
    self._medium = value
```


3 usages (2 dynamic)

```
@property  
def imageurl(self):  
    return self._imageurl
```

5 usages (2 dynamic)

```
@imageurl.setter  
def imageurl(self, value):  
    self._imageurl = value
```

3 usages (2 dynamic)

```
def printArtwork(self):  
    print("artworkid      :", self.artworkid)  
    print("artistid       :", self.artistid)  
    print("title         :", self.title)  
    print("description    :", self.description)  
    print("creationdate  :", self.creationdate)  
    print("medium          :", self.medium)  
    print("imageurl       :", self.imageurl)]
```



```
class Artist:
    def __init__(self):
        self._artistid = None
        self._name = None
        self._biography = None
        self._birthdate = None
        self._nationality = None
        self._website = None
        self._contactinformation = None
```

3 usages (2 dynamic)

@property

```
def artistid(self):
    return self._artistid
```

2 usages (2 dynamic)

@artistid.setter

```
def artistid(self, value):
    self._artistid = value
```

3 usages (2 dynamic)

@property

```
def name(self):
    return self._name
```

2 usages (2 dynamic)

@name.setter

```
def name(self, value):
    self._name = value
```

1 usage

@property

```
def biography(self):
    return self._biography
```

@biography.setter

```
def biography(self, value):
    self._biography = value
```

```
1 usage
@property
def birthdate(self):
    return self._birthdate

@birthdate.setter
def birthdate(self, value):
    self._birthdate = value

1 usage
@property
def nationality(self):
    return self._nationality

@nationality.setter
def nationality(self, value):
    self._nationality = value

1 usage
@property
def website(self):
    return self._website

@website.setter
def website(self, value):
    self._website = value

1 usage
@property
def contactinformation(self):
    return self._contactinformation

@contactinformation.setter
def contactinformation(self, value):
    self._contactinformation = value
```

```
class User:
    def __init__(self):
        self._userid = None
        self._username = None
        self._password = None
        self._email = None
        self._firstname = None
        self._lastname = None
        self._dateofbirth = None
        self._profilepicture = None
        self._favoriteartworks = []
```

1 usage

```
@property
def userid(self):
    return self._userid
```

```
@userid.setter
def userid(self, value):
    self._userid = value
```

1 usage

```
@property
def username(self):
    return self._username
```

```
@username.setter
def username(self, value):
    self._username = value
```

1 usage

```
@property
def password(self):
    return self._password
```

```
@password.setter
def password(self, value):
    self._password = value
```

```
1 usage
@property
def email(self):
    return self._email

@email.setter
def email(self, value):
    self._email = value

1 usage
@property
def firstname(self):
    return self._firstname

@firstname.setter
def firstname(self, value):
    self._firstname = value

1 usage
@property
def lastname(self):
    return self._lastname

@lastname.setter
def lastname(self, value):
    self._lastname = value

1 usage
@property
def dateofbirth(self):
    return self._dateofbirth

@dateofbirth.setter
def dateofbirth(self, value):
    self._dateofbirth = value
```

```
1 usage
@property
def profilepicture(self):
    return self._profilepicture

@profilepicture.setter
def profilepicture(self, value):
    self._profilepicture = value

1 usage
@property
def favoriteartworks(self):
    return self._favoriteartworks

@favoriteartworks.setter
def favoriteartworks(self, value):
    self._favoriteartworks = value
```

```
class Gallery:
    def __init__(self):
        self._galleryid = None
        self._name = None
        self._description = None
        self._location = None
        self._curator = None
        self._openinghours = None
        self._artworks = None
```

3 usages (2 dynamic)

```
@property
def galleryid(self):
    return self._galleryid
```

6 usages (2 dynamic)

```
@galleryid.setter
def galleryid(self, value):
    self._galleryid = value
```

3 usages (2 dynamic)

```
@property
def name(self):
    return self._name
```

6 usages (2 dynamic)

```
@name.setter
def name(self, value):
    self._name = value
```

5 usages (4 dynamic)

```
@property
def description(self):
    return self._description
```

8 usages (4 dynamic)

```
@description.setter
def description(self, value):
    self._description = value
```

3 usages (2 dynamic)

```
@property
def location(self):
    return self._location
```

6 usages (2 dynamic)

```
@location.setter
def location(self, value):
    self._location = value
```

3 usages (2 dynamic)

```
@property
def curator(self):
    return self._curator
```

6 usages (2 dynamic)

```
@curator.setter
def curator(self, value):
    self._curator = value
```

3 usages (2 dynamic)

```
@property
def openinghours(self):
    return self._openinghours
```

6 usages (2 dynamic)

```
@openinghours.setter
def openinghours(self, value):
    self._openinghours = value
```

3 usages (2 dynamic)

```
@property
def artworks(self):
    return self._artworks
```

5 usages (2 dynamic)

```
@artworks.setter
def artworks(self, value):
    self._artworks = value
```


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

}

```
class IVirtualArtGallery:
    def addArtwork(self, artwork):
        pass

    def updateArtwork(self, artwork):
        pass

    def removeArtwork(self, artworkID):
        pass

    def getArtworkById(self, artworkID):
        pass

    def searchArtworks(self, keyword):
        pass

    def addArtworkToFavorite(self, userId, artworkId):
        pass

    def removeArtworkFromFavorite(self, userId, artworkId):
        pass

    def getUserFavoriteArtworks(self, userId):
        pass
```

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 file containing connection details like hostname, dbname, username,

password, port number and returns a connection string

```
import mysql.connector
from configparser import ConfigParser
import os
```

5 usages

```
class DBConnection:
    connection = None
```

1 usage

```
@staticmethod
```

```
def getConnection():
```

```
    if not DBConnection.connection:
```

```
        config = PropertyUtil.get_config()
```

```
        connection_string = PropertyUtil.get_connection_string(config)
```

```
        DBConnection.connection = mysql.connector.connect(**connection_string)
```

```
    return DBConnection.connection
```

```

class PropertyUtil:
    1 usage
    @staticmethod
    def get_config():
        config = ConfigParser()
        thisfolder = os.path.dirname(os.path.abspath(__file__))
        dbfile = os.path.join(thisfolder, 'Database Properties')

        config.read(dbfile)
        return config

    1 usage
    @staticmethod
    def get_connection_string(config):
        hostname = config.get('Database', 'hostname')
        dbname = config.get('Database', 'dbname')
        username = config.get('Database', 'username')
        password = config.get('Database', 'password')

        # Construct the connection string
        return {
            'host': hostname,
            'user': username,
            'password': password,
            'db': dbname,
        }

```

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

```
10 usages
class CrimeAnalysisServiceImpl(IVirtualArtGallery):
    def __init__(self):
        self.connection = DBConnection.getConnection()
        self.cursor = self.connection.cursor()
```

2. Provide implementation for all the methods in the interface.

```
def showArt(self):
    q = "select * from artwork"
    try:
        self.cursor.execute(q)
        print(self.cursor.fetchall())
        self.connection.commit()
        print('Success', 'Artwork showed successfully')
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")

def addArtwork(self, artwork):
    q = "insert into artwork (ArtworkID, ArtistID, Title, Description, CreationDate, Medium, ImageURL) values (%s, %s, %s, %s, %s, %s, %s);"
    values = [artwork.artworkid, artwork.artistid, artwork.title, artwork.description, artwork.creationdate, artwork.medium, artwork.imageurl]
    try:
        self.cursor.execute(q, values)
        self.connection.commit()
        print('Success', 'Artwork added successfully')
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")

1 usage
def updateArtwork(self, artwork):
    q = "UPDATE artwork SET ArtistID = %s, Title = %s, Description = %s, CreationDate = %s, Medium = %s, ImageURL = %s WHERE ArtworkID = %s;"
    values = [artwork.artistid, artwork.title, artwork.description, artwork.creationdate, artwork.medium, artwork.imageurl, artwork.artworkid]
    try:
        self.cursor.execute(q, values)
        self.connection.commit()
        print('Success', 'Artwork updated successfully')
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False
    return True
```

```

def removeArtwork(self, artworkID):
    q = f"delete from artwork WHERE ArtworkID = {artworkID};"
    try:
        self.cursor.execute(q)
        self.connection.commit()
        print('Success', "Artwork deleted successfully")
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")

def getArtworkById(self, artworkID):
    q = f"select * from artwork WHERE ArtworkID = {artworkID};"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise ArtWorkNotFoundException
        print('Success', "Artwork deleted successfully")
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False

artwork = Artwork()
artwork.artworkid = res[0][0]
artwork.artistid = res[0][1]
artwork.title = res[0][2]
artwork.description = res[0][3]
artwork.creationdate = res[0][4]
artwork.medium = res[0][5]
artwork.imageurl = res[0][6]
return artwork

```



```
def searchArtworks(self, keyword):
    q = f"select * from artwork WHERE Title like '%{keyword}%';"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise ArtWorkNotFoundException
        print('Success', "Keyword searched successfully")
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return [], False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return [], False
    art = []
    for i in res:
        artwork = Artwork()
        artwork.artworkid = i[0]
        artwork.artistid = i[1]
        artwork.title = i[2]
        artwork.description = i[3]
        artwork.creationdate = i[4]
        artwork.medium = i[5]
        artwork.imageurl = i[6]
        art.append(artwork)
    return art, True
```

```

def addArtworkToFavorite(self, userId, artworkId):
    q = f"select FavoriteArtworks from user WHERE UserID = {userId};"
    q1 = f"select ArtworkID from Artwork"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise UserNotFoundException
        print('Success', "FavoriteArtworks found successfully")
        self.cursor.execute(q1)
        res1 = self.cursor.fetchall()
        l = []
        for i in res1:
            l.append(int(i[0]))
        if artworkId not in l:
            raise ArtWorkNotFoundException
    except UserNotFoundException:
        print("User not found")
        return False
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return None
    fav = list(map(int, res[0][0].split(",")))
    if artworkId in fav:
        print("Artwork already found in FavoriteArtworks")
        return False
    else:
        fav.append(artworkId)

```

```
f = ",".join(map(str, fav))
q = f"update user set FavoriteArtworks = '{f}' where UserID = {userId};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "FavoriteArtworks added successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")
```

```
q = f"UPDATE artwork SET NoOfFavorites = COALESCE(NoOfFavorites, 0) + 1 WHERE ArtworkID = {artworkId};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "FavoriteArtworks added successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")
```

```

def removeArtworkFromFavorite(self, userId, artworkId):
    q = f"select FavoriteArtworks from user WHERE UserID = {userId};"
    q1 = f"select ArtworkID from Artwork"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise UserNotFoundException
        self.cursor.execute(q1)
        res1 = self.cursor.fetchall()
        l = []
        for i in res1:
            l.append(int(i[0]))
        if artworkId not in l:
            raise ArtWorkNotFoundException
        print('Success', "FavoriteArtworks found successfully")
    except UserNotFoundException:
        print("User not found")
        return False
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return None
    fav = list(map(int, res[0][0].split(",")))
    if artworkId in fav:
        fav.remove(artworkId)
    else:
        print("Artwork not found in FavoriteArtworks")
        return False

```

```

f = ",".join(map(str, fav))
q = f"update user set FavoriteArtworks = '{f}' where UserID = {userId};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "FavoriteArtworks removed successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")

```

```
q = f"UPDATE artwork SET NoOfFavorites = NoOfFavorites - 1 WHERE ArtworkID = {artworkId};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "FavoriteArtworks removed successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")
```

```
def getUserFavoriteArtworks(self, userId):
    q = f"select FavoriteArtworks from user WHERE UserID = {userId};"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise UserNotFoundException
        print('Success', "FavoriteArtworks found successfully")
    except UserNotFoundException:
        print("User not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return None
    fav = list(map(int, res[0][0].split(",")))
    return fav
```



```
def searchGalleries(self, keyword):
    q = f"select * from gallery WHERE Name like '{keyword}%'";
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise GalleryNotFoundException
        print('Success', "Keyword searched successfully")
    except GalleryNotFoundException:
        print("Gallery not found")
        return [], False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return [], False
    gal = []
    for i in res:
        gallery = Gallery()
        gallery.galleryid = i[0]
        gallery.name = i[1]
        gallery.description = i[2]
        gallery.location = i[3]
        gallery.curator = i[4]
        gallery.openinghours = i[5]
        gal.append(gallery)
    return gal, True
```

```

def addArtworkToGallery(self, galleryId, artworkId):
    q = f"select Artworks from gallery WHERE GalleryID = {galleryId};"
    q1 = f"select ArtworkID from Artwork"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise GalleryNotFoundException
        print('Success', "Artworks found successfully")
        self.cursor.execute(q1)
        res1 = self.cursor.fetchall()
        l = []
        for i in res1:
            l.append(int(i[0]))
        if artworkId not in l:
            raise ArtWorkNotFoundException
    except GalleryNotFoundException:
        print("Gallery not found")
        return False
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return None
    fav = list(map(int, res[0][0].split(",")))
    if artworkId in fav:
        print("Artwork already found in Gallery")
        return False
    else:
        fav.append(artworkId)

```



```
f = ",".join(map(str, fav))
q = f"update gallery set Artworks = '{f}' where GalleryID = {galleryId};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "Artworks added successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")
return True
```

Usage

```
def removeArtworkFromGallery(self, galleryId, artworkId):
    q = f"select Artworks from gallery WHERE GalleryID = {galleryId};"
    q1 = f"select ArtworkID from Artwork"
    res = None
    try:
        self.cursor.execute(q)
        res = self.cursor.fetchall()
        self.connection.commit()
        if len(res) == 0:
            raise GalleryNotFoundException
        print('Success', "Artworks found successfully")
        self.cursor.execute(q1)
        res1 = self.cursor.fetchall()
        l = []
        for i in res1:
            l.append(int(i[0]))
        if artworkId not in l:
            raise ArtWorkNotFoundException
    except GalleryNotFoundException:
        print("Gallery not found")
        return False
    except ArtWorkNotFoundException:
        print("Artwork not found")
        return False
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False
    fav = list(map(int, res[0][0].split(",")))
    if artworkId in fav:
        fav.remove(artworkId)
    else:
        print("Artwork not found in Gallery")
        return False
```

```

f = ",".join(map(str, fav))
q = f"update gallery set Artworks = '{f}' where GalleryID = {galleryID};"
try:
    self.cursor.execute(q)
    self.connection.commit()
    print('Success', "Artworks removed successfully")
except Exception as e:
    print("The exception is:", e)
    print("Error", "Trouble adding data into Database")
    return False
return True

```

```

def addGallery(self, gallery):
    q = "insert into gallery (GalleryID, Name, Description, Location, Curator, OpeningHours, Artworks) values (%s, %s, %s, %s, %s, %s, %s);"
    values = [gallery.galleryid, gallery.name, gallery.description, gallery.location, gallery.curator, gallery.openinghours, gallery.artworks]
    try:
        self.cursor.execute(q, values)
        self.connection.commit()
        print('Success', "Gallery added successfully")
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False
    return True

! usage
def updateGallery(self, gallery):
    q = "UPDATE gallery SET Name = %s, Description = %s, Location = %s, Curator = %s, OpeningHours = %s, Artworks = %s WHERE GalleryID = %s;"
    values = [gallery.name, gallery.description, gallery.location, gallery.curator, gallery.openinghours, gallery.artworks, gallery.galleryid]
    try:
        self.cursor.execute(q, values)
        self.connection.commit()
        print('Success', "Gallery updated successfully")
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False
    return True

! usage
def removeGallery(self, galleryid):
    q = f"delete from gallery WHERE GalleryID = {galleryid};"
    try:
        self.cursor.execute(q)
        self.connection.commit()
        print('Success', "Gallery deleted successfully")
    except Exception as e:
        print("The exception is:", e)
        print("Error", "Trouble adding data into Database")
        return False
    return True

```

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

```
12 usages
class ArtWorkNotFoundException(Exception):
    pass

6 usages
class GalleryNotFoundException(Exception):
    pass

6 usages
class UserNotFoundException(Exception):
    pass
```

9. Main Method

Create class named MainModule with main method in main package.

Trigger all the methods in service implementation class.

```
1 usage
class MainModule:
    1 usage
    @staticmethod
    def main():
        service = CrimeAnalysisServiceImpl()
        while True:
            print("")
            print("Menu \n")
            print("1. Add Artwork")
            print("2. Show Arts")
            print("3. Show Artists")
            print("4. Update Artwork")
            print("5. Remove Artwork")
            print("6. Get Artwork by ID")
            print("7. Search using Keyword")
            print("8. Add to Favorite")
            print("9. Remove from Favorite")
            print("10. get user Favorites")
            print("11. Exit")
            print("")
            choice = int(input("Enter your choice: "))
            if choice < 1 or choice > 11:
                print("enter valid option")
                continue
```

```
if choice == 1:
    artwork = Artwork()
    artwork.artistid = int(input("Enter Artist ID :"))
    artwork.title = input("Enter Title :")
    artwork.description = input("Enter Description :")
    Creationdate = input("Enter Creationdate (YYYY-MM-DD): ")
    user_date = datetime.strptime(Creationdate, __format: "%Y-%m-%d").date()
    artwork.creationdate = user_date
    artwork.medium = input("Enter Medium :")
    artwork.imageurl = input("Enter ImageURL :")

    service.addArtwork(artwork)

elif choice == 2:
    service.showart()
elif choice == 3:
    service.showartist()
elif choice == 4:
    artwork = Artwork()
    artwork.artistid = int(input("Enter Artist ID :"))
    artwork.title = input("Enter Title :")
    artwork.description = input("Enter Description :")
    Creationdate = input("Enter Creationdate (YYYY-MM-DD): ")
    user_date = datetime.strptime(Creationdate, __format: "%Y-%m-%d").date()
    artwork.creationdate = user_date
    artwork.medium = input("Enter Medium :")
    artwork.imageurl = input("Enter ImageURL :")

    service.updateArtwork(artwork)
```

```
elif choice == 5:
    service.removeArtwork(int(input("Enter Artwork ID to remove :")))
elif choice == 6:
    artwork = service.getArtworkById(int(input("Enter Artwork ID :")))
    if artwork:
        artwork.printArtwork()
elif choice == 7:
    ser, bool = service.searchArtworks(input("Enter keyword to search :"))
    for i in ser:
        i.printArtwork()
        print("")
elif choice == 8:
    service.addArtworkToFavorite(int(input("Enter User ID :")), int(input("Enter Artwork ID :")))
elif choice == 9:
    service.removeArtworkFromFavorite(int(input("Enter User ID :")), int(input("Enter Artwork ID :")))
elif choice == 10:
    fav = service.getUserFavoriteArtworks(int(input("Enter User ID :")))
    print(fav)
elif choice == 11:
    break
```

```
if __name__ == "__main__":
    MainModule.main()
```


addArtwork();

Output :

```
Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 1
Enter Artist ID :3
Enter Title :def
Enter Description :descri
Enter Creationdate (YYYY-MM-DD): 2024-05-28
Enter Medium :oil paint
Enter ImageURL :nan
Success Artwork added successfully
```

updateArtwork();

Output :

```
Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 4
Enter Artwork ID :12
Enter Artist ID :3
Enter Title :dsa
Enter Description :dsa
Enter Creationdate (YYYY-MM-DD): 2024-05-20
Enter Medium :qwe
Enter ImageURL :qwe
Success Artwork updated successfully
```

removeArtwork()

Output :

```
Enter your choice: 5
Enter Artwork ID to remove :12
Success Artwork deleted successfully
```

getArtworkById()

Output :

```
Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 6
Enter Artwork ID :3
Success Artwork fetched successfully
artworkid      : 3
artistid       : 1
title          : Vitruvian Man
description     : A study of the proportions of the human body based on Vitruvius' principles.
creationdate    : 2001-08-24
medium         : Pen and ink with wash over metalpoint on paper
imageurl       : https://leonardodavinci.net/images/drawings/vitruvian-man.jpg
```

searchArtworks();

Output :

```
Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 7
Enter keyword to search :the
Success Keyword searched successfully
artworkid      : 2
artistid       : 1
title          : The Last Supper
description     : A depiction of the Last Supper of Jesus with his disciples.
creationdate    : 2002-01-04
medium         : Fresco
imageurl       : https://leonardodavinci.net/images/paintings/the-last-supper.jpg

artworkid      : 6
artistid       : 2
title          : The Bedroom
description     : A depiction of Van Gogh's bedroom in the Yellow House in Arles, France.
creationdate    : 2002-06-22
medium         : Oil on canvas
imageurl       : https://www.vangoghmuseum.nl/-/media/images/vgm/studie-web/uitgelicht/the-bedroom.jpg

artworkid      : 7
artistid       : 3
title          : The Two Fridas
description     : A double self-portrait depicting two different aspects of Frida's personality.
creationdate    : 2001-06-06
medium         : Oil on canvas
imageurl       : https://www.fridakahlo.org/images/paintings/the-two-fridas.jpg

artworkid      : 10
artistid       : 3
title          : The Broken Column
description     : A self-portrait depicting Frida Kahlo with a broken column representing her spinal injuries.
creationdate    : 2001-05-11
medium         : Oil on canvas
imageurl       : https://www.fridakahlo.org/images/paintings/the-broken-column.jpg
```

addArtworkToFavorite();

Output :

Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 8

Enter User ID :1

Enter Artwork ID :8

Success FavoriteArtworks found successfully

Success FavoriteArtworks added successfully

Success FavoriteArtworks added successfully

removeArtworkFromFavorite();

Output :

```
Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 9
Enter User ID :1
Enter Artwork ID :8
Success FavoriteArtworks found successfully
Success FavoriteArtworks removed successfully
Success FavoriteArtworks removed successfully
```

showFavoritesofUser();

Output :

Menu

1. Add Artwork
2. Show Arts
3. Show Artists
4. Update Artwork
5. Remove Artwork
6. Get Artwork by ID
7. Search using Keyword
8. Add to Favorite
9. Remove from Favorite
10. get user Favorites
11. Exit

Enter your choice: 10

Enter User ID :1

Success FavoriteArtworks found successfully

[1, 2, 4, 5]

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.

```
import pytest
from DAO.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from Entity.Artwork import Artwork
from Entity.Gallery import Gallery

def test_addArtworkToGallery():
    service = CrimeAnalysisServiceImpl()
    assert service.addArtworkToGallery(galleryId: 1, artworkId: 8) == True
```

b. Verify that updating artwork details works correctly.

```
def test_updateArtwork():
    service = CrimeAnalysisServiceImpl()
    artwork = Artwork()
    artwork.artworkid = 1
    artwork.artistid = 1
    artwork.title = 'Mona Lisa'
    artwork.description = 'The famous portrait of a woman with an enigmatic smile.'
    artwork.creationdate = '2001-05-20'
    artwork.medium = 'Oil on poplar panel'
    artwork.imageurl = 'https://leonardodavinci.net/images/paintings/mona-lisa.jpg'
    assert service.updateArtwork(artwork) == True
```

c. Test removing an artwork from the gallery.

```
def test_removeArtworkFromGallery():
    service = CrimeAnalysisServiceImpl()
    assert service.removeArtworkFromGallery(galleryId: 1, artworkId: 8) is True
```

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

```
def test_searchArtworks():
    service = CrimeAnalysisServiceImpl()
    art, bool = service.searchArtworks("Fridas")
    assert bool == True
```

2. Gallery Management:

- a. Test creating a new gallery.

```
def test_addGallery():
    service = CrimeAnalysisServiceImpl()
    gallery = Gallery()
    gallery.galleryid = 4
    gallery.name = "123 gal"
    gallery.description = 'The famous portrait of a woman with an enigmatic smile.'
    gallery.location = 'chennai0'
    gallery.curator = 1
    gallery.openinghours = "Monday to Friday: 10 AM - 6 PM, Saturday: 12 PM - 4 PM"
    gallery.artworks = "1,3"
    assert service.addGallery(gallery) == True
```

- b. Verify that updating gallery information works correctly.

```
def test_updateGallery():
    service = CrimeAnalysisServiceImpl()
    gallery = Gallery()
    gallery.galleryid = 4
    gallery.name = "123 gal"
    gallery.description = 'The famous portrait of a woman with an enigmatic smile.'
    gallery.location = 'bangalore'
    gallery.curator = 1
    gallery.openinghours = "Monday to Friday: 10 AM - 6 PM, Saturday: 12 PM - 4 PM"
    gallery.artworks = "1,3"
    assert service.updateGallery(gallery) == True
```

c. Test removing a gallery from the system.

```
def test_removeGallery():  
    service = CrimeAnalysisServiceImpl()  
    assert service.removeGallery(4) == True
```

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

```
def test_searchGalleries():  
    service = CrimeAnalysisServiceImpl()  
    art, bool = service.searchGalleries("Gallery")  
    assert bool == True
```

Test Results :

```
===== test session starts =====  
collecting ... collected 8 items  
  
Test_Ant_Gallery.py::test_addArtworkToGallery PASSED [ 12%]Connected to the database  
Success Artworks found successfully  
Success Artworks added successfully  
  
Test_Ant_Gallery.py::test_updateArtwork PASSED [ 25%]Success Artwork updated successfully  
  
Test_Ant_Gallery.py::test_removeArtworkFromGallery PASSED [ 37%]Success Artworks found successfully  
Success Artworks removed successfully  
  
Test_Ant_Gallery.py::test_searchArtworks PASSED [ 50%]Success Keyword searched successfully  
  
Test_Ant_Gallery.py::test_addGallery PASSED [ 62%]Success Gallery added successfully  
  
Test_Ant_Gallery.py::test_updateGallery PASSED [ 75%]Success Gallery updated successfully  
  
Test_Ant_Gallery.py::test_removeGallery PASSED [ 87%]Success Gallery deleted successfully  
  
Test_Ant_Gallery.py::test_searchGalleries PASSED [100%]Success Keyword searched successfully  
  
===== 8 passed in 0.25s =====  
  
Process finished with exit code 0
```