Case Study: Virtual Art Gallery

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)

```
class Artwork(vag):
    def __init__(self, Title, Description, CreationDate, Medium, ImageURL):
        self.Title = Title
        self.Description = Description
        self.CreationDate = CreationDate
        self.Medium = Medium
        self.ImageURL = ImageURL
```

• Artist

ArtistID (Primary Key)

Name

Biography

BirthDate

Nationality

Website

Contact Information

```
def
__init__(self,ArtistID,Name,Biography,BirthDate,Nationality,Website,Contact
Info):
    self.ArtistID=ArtistID
    self.Name=Name
    self.Biography=Biography
    self.BirthDate=BirthDate
    self.Nationality=Nationality
```

```
    self.Website=Website
    self.ContactInfo=ContactInfo
```

• User

UserID (Primary Key)

Username

Password

Email

First Name

Last Name

Date of Birth

Profile Picture

FavoriteArtworks (a list of references to ArtworkIDs)

```
class user(vag):
    def
__init__(self,Username,Password,Email,FirstName,LastName,DOB,ProfilePic,Fav
oriteArtworks):
        self.Username=Username,
        self.Password=Password,
        self.Email=Email,
        self.FirstName=FirstName,
        self.LastName=LastName,
        self.DOB=DOB
        self.ProfilePic=ProfilePic,
        self.FavoriteArtworks=FavoriteArtworks
```

• Gallery

GalleryID (Primary Key)

Name

Description

Location

Curator (Reference to ArtistID)

OpeningHours

```
def
__init__(self,GalleryID,Name,Description,Location,Curator,OpeningHours):
    self.GalleryID=GalleryID,
    self.Name=Name,
    self.Description=Description,
    self.Location=Location,
    self.Curator=Curator,
    self.OpeningHours=OpeningHours
```

Back-End Design

```
mysql> use artgallerycasestudy;
Database changed
mysql> show tables;
+-------------------------------+
| Tables_in_artgallerycasestudy |
+-------------------------------+
| artist                        |
| artwork                       |
| gallery                       |
| user                          |
+-------------------------------+
4 rows in set (0.02 sec)
```

Tables:

```
mysql> desc artist;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ArtistID    | int          | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(40)  | YES  |     | NULL    |                |
| Biography   | text         | YES  |     | NULL    |                |
| DOB         | date         | YES  |     | NULL    |                |
| Nationality | varchar(30)  | YES  |     | NULL    |                |
| Website     | varchar(100) | YES  |     | NULL    |                |
| ContactInfo | varchar(40)  | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
7 rows in set (0.02 sec)
```

```
mysql> desc artwork;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| ArtworkID    | int          | NO   | PRI | NULL    | auto_increment |
| Title        | varchar(40)  | YES  |     | NULL    |                |
| Description  | text         | YES  |     | NULL    |                |
| CreationDate | date         | YES  |     | NULL    |                |
| Medium       | varchar(40)  | YES  |     | NULL    |                |
| ImageUrl     | varchar(100) | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

```
mysql> desc gallery;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| GalleryID    | int          | NO   | PRI | NULL    | auto_increment |
| Name         | varchar(40)  | YES  |     | NULL    |                |
| Description  | varchar(100) | YES  |     | NULL    |                |
| Location     | varchar(50)  | YES  |     | NULL    |                |
| Curator      | int          | YES  | MUL | NULL    |                |
| OpeningHours | time         | YES  |     | NULL    |                |
| ClosingHours | time         | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

```
mysql> desc user;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| UserID           | int          | NO   | PRI | NULL    | auto_increment |
| Username         | varchar(40)  | YES  |     | NULL    |                |
| Password         | varchar(40)  | YES  |     | NULL    |                |
| Email            | varchar(40)  | YES  |     | NULL    |                |
| FirstName        | varchar(40)  | YES  |     | NULL    |                |
| LastName         | varchar(40)  | YES  |     | NULL    |                |
| DOB              | date         | YES  |     | NULL    |                |
| ProfilePic       | varchar(100) | YES  |     | NULL    |                |
| FavoriteArtworks | varchar(50)  | YES  |     | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Coding Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters) Service Provider Interface/Abstract class Keep the interfaces and implementation classes in package dao Create IVirtualArtGallery Interface/abstract class with the following methods

```python
from abc import ABC,abstractmethod

class VirtualArtGallery(ABC):
    @abstractmethod
    def addArtWork(self):
        pass
    def updateArtWork(self):
        pass
    def removeArtWork(self):
        pass
    def getArtworkById(self):
        pass
    def searchArtWorks(self):
        pass
    def addArtworkToFavorites(self,id,id2):
        pass
    def getUserFavoriteArtworks(self):
        pass
    def removeArtworkFromFavourites(self):
        pass
```

// Artwork Management

addArtwork(); parameters- Artwork object return type Boolean

```
def addArtWork(self, a):
    cursor = d.connection.cursor()
    try:
        query = 'insert into
Artwork(Title,Description,CreationDate,Medium,ImageURL)
values(%s,%s,%s,%s,%s)'
        data = (a.Title, a.Description, a.CreationDate, a.Medium,
a.ImageURL)
        a2=f'select * from artwork where Title=%s'
        data2=(a.Title,)
        cursor.execute(a2, data2)
        if cursor.fetchone():
            raise ex.ArtworkAlreadyExists("this artwork already exists")
            cursor.close()
        else:
            cursor.execute(query, data)
            d.connection.commit()
            print("data entered")
            cursor.close()
    except ex.ArtworkAlreadyExists as e:
        pass
```

o/P:

```
1
Enter title :shiva
enter description : the great MARATHA KING
Enter creation date in 'YYYY-mm-dd' format : 2010-05-06
enter medium : marathi
enter image url : www.shivaji.com
data entered
```

Entering the artwork again:

```
1
Enter title :shiva
TitleAlreadyExistsException :Title Title Already Exists can't add to artwork
```

updateArtwork();

```
def updateArtWork(self,Title,column_name,new_value):
    cursor= d.connection.cursor()
    try:
        q=f'select * from artwork where Title=%s'
        data=(Title,)
        cursor.execute(q,data)
        if cursor.fetchone():
            q2=f'update artwork set {column_name} =%s where Title=%s'
            d2=(new_value,Title)
            cursor.execute(q2,d2)
            d.connection.commit()
```

```
            cursor.close()
            print("Values updated in artwork gallery")
        else:
            raise ex.ArtworkNotFoundException("The artwork is not found")
    except ex.ArtworkNotFoundException as e:
        pass
```

o/p:

```
2
enter title of the art work : raja
Enter column name which is to be updated : Medium
enter new value : English
Values updated in artwork gallery
```

```
2
enter title of the art work : kingu
TitleNotFoundException :The title kingu doesn't exists
```

removeArtwork() parameters-artworkID return type Boolean

```
def removeArtWork(self,ID):
    cursor=d.connection.cursor()
    try:
        q='select * from artwork where ArtworkID=%s'
        data=(ID,)
        cursor.execute(q,data)
        if cursor.fetchone():
            query=f'delete from artwork where ArtworkID={ID}'
            cursor.execute(query)
            print(f'Data corresponding to artworkID:{ID} is removed')
            d.connection.commit()
            cursor.close()
        else:
            raise ex.ArtworkNotFoundException(f"The artwork with id:{ID}
doesn't exists")
    except ex.ArtworkNotFoundException as e:
        pass
```

o/p:

```
3
Please Enter Id of Artwork to be removed: 8
Data corresponding to artworkID:8 is removed
```

```
3
Please Enter Id of Artwork to be removed: 1
ArtworkNotFoundException: The artwork with id:1 doesn't exists
```

getArtworkById(); parameters-artworkID return type Artwork

```
def getArtworkById(self,ID):
    cursor = d.connection.cursor()
    try:
        q = 'select * from artwork where ArtworkID=%s'
        data = (ID,)
        cursor.execute(q, data)
        if cursor.fetchone():
            query = f'select * from artwork where ArtworkID={ID}'
            cursor.execute(query)
            answer=cursor.fetchall()
            print("ArtWorkId | Title | Description | Creation Date | Medium
| Image Url")
            for x in answer:
                print(x)
            cursor.close()
        else:
            raise ex.ArtworkNotFoundException(f"The artwork with id:{ID}
doesn't exists")
    except ex.ArtworkNotFoundException as e:
        pass
```

o/p:

```
4
enter artwork id : 5
ArtWorkId | Title | Description | Creation Date | Medium | Image Url
(5, 'raja', 'raja the great', datetime.date(2022, 5, 10), 'English', 'sss')
```

```
4
enter artwork id : 8
ArtworkNotFoundException: The artwork with id:8 doesn't exists
```

searchArtworks(); parameters- keyword return type list of Artwork Object

```
def searchArtWorks(self, keyword):
    cursor = d.connection.cursor()
    try:
        q = f"select * from artwork where Description like '%{keyword}%'
or Title like '%{keyword}%'  or Medium like '%{keyword}%' or ImageUrl like
'%{keyword}%'"
        cursor.execute(q)
        result = cursor.fetchall()
        if result:
            print("ArtWorkId | Title | Description | Creation Date | Medium
| Image Url")
            for x in result:
                print(x)
```

```
            cursor.close()
        else:
            raise ex.ArtworkNotFoundException(f" No Art Work Found")
    except ex.ArtworkNotFoundException as e:
        pass
```

o/p:

```
5
Enter key word : great
ArtWorkId | Title | Description | Creation Date | Medium | Image Url
(5, 'raja', 'raja the great', datetime.date(2022, 5, 10), 'English', 'sss')
(9, 'greatKing', 'alexandar ', datetime.date(1982, 4, 2), 'Parsi', 'WWW.alexandar.com')
```

addArtworkToFavorite(); parameters- userId, artworkId return type Boolean

```
def
addUser(self,Username,Password,Email,FirstName,LastName,DOB,ProfilePic,Favo
riteArtworks):
    cursor = d.connection.cursor()
    try:
        query = 'insert into
user(Username,Password,Email,FirstName,LastName,DOB,ProfilePic,FavoriteArtw
orks) values(%s,%s,%s,%s,%s,%s,%s,%s)'
        data =
(Username,Password,Email,FirstName,LastName,DOB,ProfilePic,FavoriteArtworks
)
        a2=f'select * from user where Username=%s'
        data2=(Username,)
        cursor.execute(a2, data2)
        if cursor.fetchone():
            raise ex.UserAlreadyExists("User already exists")
            cursor.close()
        else:
            cursor.execute(query, data)
            d.connection.commit()
            print("data entered")
            cursor.close()
    except ex.UserAlreadyExists as e:
        pass
```

o/p:

```
7
please enter user ID: 4
please enter artwork ID: 9
values updated
```

removeArtworkFromFavorite() parameters- userId, artworkId return type Boolean

```
def removeArtworkFromFavourites(self,UserID,ArtworkID):
    cursor = d.connection.cursor()
    try:
        q1 = f"select * from User where UserID={UserID}"
```

```
        cursor.execute(q1)
        if cursor.fetchone():
            try:
                q2 = f"select * from artwork where ArtworkID={ArtworkID}"
                cursor.execute(q2)
                if cursor.fetchone():
                    q3 = f"select FavoriteArtworks from user  where
UserID={UserID}"
                    cursor.execute(q3)
                    ans = cursor.fetchone()
                    if ans:
                        favorites = ans[0]
                        if favorites:
                            artworks_list = favorites.split(',')
                            if str(ArtworkID) in artworks_list:
                                artworks_list.remove(str(ArtworkID))
                                new_favorite_artworks =
','.join(artworks_list)
                                q4 = "UPDATE user SET FavoriteArtworks = %s
WHERE UserID = %s"
                                cursor.execute(q4, (new_favorite_artworks,
UserID))
                                d.connection.commit()
                                print(f"Artwork with ID {ArtworkID} removed
from favourites")
                            else:
                                print("Artwork not found in favorites")
                else:
                    raise ex.ArtworkNotFoundException("Art work not found
")
            except ex.ArtworkNotFoundException as e:
                pass
        else:
            raise ex.UserNotFoundException(f"User with  ID {UserID} not
found ")
    except ex.UserNotFoundException as e:
        pass
```

```
8
please enter user ID: 4
please enter artwork ID: 9
Artwork with ID 9 removed from favourites
```

```
8
please enter user ID: 2
please enter artwork ID: 9
Artwork not found in favorites
```

getUserFavoriteArtworks() parameters- userId return type Boolean

```
def getUserFavoriteArtworks(UserId):
    cursor=d.connection.cursor()
    try:
        q="select Username,FavoriteArtworks from user where UserID=%s"
        data=(userId,)
        cursor.execute(q,data)
        x=cursor.fetchone()
        if x:

print("Username,Password,Email,FirstName,LastName,DOB,ProfilePic,FavoriteAr
tworks")
             for i in x:
                print(i)
        else:
            raise ex.UserNotFoundException(f"The user with userID
{userId}")
    except ex.UserNotFoundException as e:
        pass
```

o/P:

```
9
please enter user ID: 4
Username,FavoriteArtworks
('prasad', '5,9')
```

```
class DBConnection:
    connection = None

    @staticmethod
    def getConnection():
        if self.connection is None:
            connection_string = PropertyUtil.getPropertyString()
            self.connection = mysql.connector.connect(**connection_string)
        return self.connection


class PropertyUtil:
    @staticmethod
    def getPropertyString():
        config = configparser.ConfigParser()
        config.read('connection.properties')
        connection_properties = {
            'host': config.get('Connection', 'hostname'),
            'database': config.get('Connection', 'dbname'),
            'user': config.get('Connection', 'username'),
            'password': config.get('Connection', 'password'),
            'port': config.getint('Connection', 'port')
        }
        return connection_properties
```

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

```
class ArtworkNotFoundException(Exception):
    def __init__(self,message):
        self.message=message
        print(f"ArtworkNotFoundException: {self.message}")
```

This exception is used in different methods

Like

```
def searchArtWorks(self, keyword):
def getArtworkById(self,ID):
def removeArtWork(self,ID):
```

2. UserNotFoundException :throw this exception when user enters an invalid id which doesn't

exist in db

```
class UserNotFoundException(Exception):
    def __init__(self,message):
        self.message=message
        print(f"UserNotFoundException: {self.message}")
```

This exception is used in different methods like

```
def removeArtworkFromFavourites(self,UserID,ArtworkID):
def addArtworkToFavorites(self,UserID,ArtworkID):
def getUserFavoriteArtworks(UserId):
```

9. Main Method Create class named MainModule with main method in main package. Trigger all the methods in service implementation class

Menu Display

```
while(True):
    print("\n"
          "Enter A Choice \n"
          "1.Add ArtWork \n"
          "2.Update values in Artwork \n"
          "3.remove Artwork \n"
          "4.display the artwork \n"
          "5.search art work with key word \n"
          "6.add user \n"
```

```
          "7.add artwork to favourites \n"
          "8.remove art work from favorites \n"
          "9.get favorite artworks of the user \n")
    choice=int(input())
    if choice==1:
        title=input("Enter title :")
        try:
            x=a.checkIfTitleExists(title)
            if x:
                ex.TitleAlreadyExists("Title Title Already Exists can't add
to artwork")
            else:
                description=input("enter description : ")
                creation_date=input("Enter creation date in 'YYYY-mm-dd'
format : ")
                medium=input("enter medium : ")
                image_url=input("enter image url : ")

a.addArtWork(title,description,creation_date,medium,image_url)
                print("\n")
        except ex.TitleAlreadyExists as e:
            pass

    elif choice==2:
        title=input("enter title of the art work : ")
        try:
            k=a.checkIfTitleExists(title)
            if k:
                column_name=input("Enter column name which is to be updated
: ")
                new_value=input("enter new value : ")
                a.updateArtWork(title,column_name,new_value)
                print("\n")
            else:
                raise ex.TitleNotFoundException(f"The title {title} doesn't
exists")
        except ex.TitleNotFoundException as e:
            pass
    elif choice==3:
        id=int(input("Please Enter Id of Artwork to be removed: "))
        a.removeArtWork(id)
        print("\n")

    elif choice ==4:
        id=int(input("enter artwork id : "))
        a.getArtworkById(id)
        print("\n")

    elif choice==5:
        keyword=input("Enter key word : ")
        a.searchArtWorks(keyword)
        print("\n")

    elif choice==6:
         Username=input("enter user-name : ")
         try:
             if b.checkIfUsernameExists(Username):
                 raise ex.UsernameAlreadyExistsException("This user name
already exists ")
             else:
                Password=input("enter password : ")
```

```
                Email =input("enter email : ")
                FirstName=input("enter first name : ")
                LastName =input("enter last name : ")
                DOB=input("enter date of birth : ")
                ProfilePic=input("enter profile picture link : ")
                FavoriteArtworks=input("enter favourite art works :  ")

b.addUser(Username,Password,Email,FirstName,LastName,DOB,ProfilePic,Favorit
eArtworks)
                print("\n")
            except ex.UsernameAlreadyExistsException as e:
                pass


    elif choice==7:
        userID=int(input("please enter user ID: "))
        artworkID=int(input("please enter artwork ID: "))
        b.addArtworkToFavorites(userID,artworkID)
        print("\n")

    elif choice==8:
        userID = int(input("please enter user ID: "))
        artworkID = int(input("please enter artwork ID: "))
        b.removeArtworkFromFavourites(userID,artworkID)
        print("\n")

    elif choice==9:
        userID = int(input("please enter user ID: "))
        b.getUserFavoriteArtworks(userID)
        print("\n")

    elif choice==0:
        break
    else:
        print("invalid option please choose again \n")
```

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.

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

```
import pytest
from Artwork import Artwork
from User import user

@pytest.fixture
def a():
    return Artwork()
def testadding(a):
    k=a.addArtWork("Title", "Description", "2024-05-08", "Medium",
"ImageURL")
    assert k is not None

def test_remove_artwork(a):
    assert a.removeArtWork(6)==True

def testArtworkRetrival(a):
    artwork = a.getArtworkById(6)
    assert artwork is not None
    assert a.getArtworkById(1000) is False
def testSearch(a):
    k= a.searchArtWorks('king')
    assert k is not None

@pytest.fixture
def b():
    return user()
def testAddUserFromFavourites(b):
    assert b.addArtworkToFavorites(3, 7) is True
def testremoveUserFromFavourites(b):
   assert b.removeArtworkFromFavourites(3,7) is True
   assert b.removeArtworkFromFavourites(3,1) is False
```

O/p

```
✓ Tests passed: 6 of 6 tests – 15 ms

C:\Users\Lenovo\PycharmProjects\virtual_art_gallery\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2024.1/p
Testing started at 01:01 pm ...
Launching pytest with arguments C:\Users\Lenovo\PycharmProjects\virtual_art_gallery\unit_test.py --no-header --no-summary -q in C:\Users\Le

============================= test session starts =============================
collecting ... collected 6 items

unit_test.py::testadding PASSED                                      [ 16%]ArtworkAlreadyExists exception this artwork already exists

unit_test.py::test_remove_artwork PASSED                             [ 33%]ArtworkNotFoundException: The artwork with id:6 doesn't exis

unit_test.py::testArtworkRetrival PASSED                             [ 50%]ArtworkNotFoundException: The artwork with id:6 doesn't exis
ArtworkNotFoundException: The artwork with id:1000 doesn't exists

unit_test.py::testSearch PASSED                                      [ 66%]ArtworkNotFoundException:  No Art Work Found

unit_test.py::testAddUserFromFavourites PASSED                       [ 83%]values updated

unit_test.py::testremoveUserFromFavourites PASSED                    [100%]Artwork with ID 7 removed from favourites
ArtworkNotFoundException: Art work not found


============================== 6 passed in 0.12s ==============================

Process finished with exit code 0
```