Vitual Art Gallery

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

1.Artist

```
class Artist:
    def __init__(self, artistID=None, name=None, biography=None, birthDate=None,
nationality=None, website=None, contactInformation=None):
        self.__artistID = artistID
        self.__name = name
        self.__biography = biography
        self.__birthDate = birthDate
        self.__nationality = nationality
        self.__website = website
        self.__contactInformation = contactInformation

    # Getter methods
    def get_artistID(self):
        return self.__artistID

    def get_name(self):
        return self.__name

    def get_biography(self):
        return self.__biography

    def get_birthDate(self):
        return self.__birthDate

    def get_nationality(self):
        return self.__nationality

    def get_website(self):
        return self.__website

    def get_contactInformation(self):
        return self.__contactInformation

    # Setter methods
    def set_artistID(self, artistID):
        self.__artistID = artistID

    def set_name(self, name):
        self.__name = name

    def set_biography(self, biography):
        self.__biography = biography

    def set_birthDate(self, birthDate):
        self.__birthDate = birthDate

    def set_nationality(self, nationality):
        self.__nationality = nationality

    def set_website(self, website):
        self.__website = website

    def set_contactInformation(self, contactInformation):
        self.__contactInformation = contactInformation
```

2.Artwork

```
class Artwork:
    def __init__(self, artworkId=None, title=None, description=None, creationDate=None,
```

```
medium=None, imageUrl=None, artistId=None):
        self.__artworkId = artworkId
        self.__title = title
        self.__description = description
        self.__creationDate = creationDate
        self.__medium = medium
        self.__imageUrl = imageUrl


    def __str__(self):
        return f"Artwork ID: {self.__artworkId}, Title: {self.__title}, Description:
{self.__description}, Creation Date: {self.__creationDate}, Medium: {self.__medium},
Image URL: {self.__imageUrl}"

    # Getter methods
    def get_artworkId(self):
        return self.__artworkId

    def get_title(self):
        return self.__title

    def get_description(self):
        return self.__description

    def get_creationDate(self):
        return self.__creationDate

    def get_medium(self):
        return self.__medium

    def get_imageUrl(self):
        return self.__imageUrl


     # Setter methods
    def set_artworkId(self, artworkId):
        self.__artworkId = artworkId

    def set_title(self, title):
        self.__title = title

    def set_description(self, description):
        self.__description = description

    def set_creationDate(self, creationDate):
        self.__creationDate = creationDate

    def set_medium(self, medium):
        self.__medium = medium

    def set_imageUrl(self, imageUrl):
        self.__imageUrl = imageUrl
```

3.User

```
class User:
    def __init__(self, user_id, username, password, email, first_name, last_name,
date_of_birth, profile_picture, favorite_artworks=None):
        self.__user_id = user_id
        self.__username = username
        self.__password = password
        self.__email = email
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__profile_picture = profile_picture
```

```
        self.__favorite_artworks = favorite_artworks if favorite_artworks else []

    # Getters
    def get_user_id(self):
        return self.__user_id

    def get_username(self):
        return self.__username

    def get_password(self):
        return self.__password

    def get_email(self):
        return self.__email

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def get_profile_picture(self):
        return self.__profile_picture

    def get_favorite_artworks(self):
        return self.__favorite_artworks

    # Setters
    def set_username(self, username):
        self.__username = username

    def set_password(self, password):
        self.__password = password

    def set_email(self, email):
        self.__email = email

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

    def set_profile_picture(self, profile_picture):
        self.__profile_picture = profile_picture

    def add_favorite_artwork(self, artwork_id):
        self.__favorite_artworks.append(artwork_id)

    def remove_favorite_artwork(self, artwork_id):
        if artwork_id in self.__favorite_artworks:
            self.__favorite_artworks.remove(artwork_id)
```

4.Gallery

```
class Gallery:
    def __init__(self, galleryID=None, name=None, website=None, description=None,
location=None, curator=None, opening_hours=None):
        self.__galleryID = galleryID
        self.__name = name
        self.__website = website
        self.__description = description
```

```
        self.__location = location
        self.__curator = curator
        self.__opening_hours = opening_hours

    def __str__(self):
            return f"Gallery ID: {self.__galleryID}, Name: {self.__name}, Website:
{self.__website}, Description: {self.__description}, Location: {self.__location},
Curator: {self.__curator}, Opening Hours: {self.__opening_hours}"

    # Getter methods
    def get_galleryID(self):
        return self.__galleryID

    def get_name(self):
        return self.__name

    def get_website(self):
        return self.__website

    def get_description(self):
        return self.__description

    def get_location(self):
        return self.__location

    def get_curator(self):
        return self.__curator

    def get_opening_hours(self):
        return self.__opening_hours

    # Setter methods
    def set_galleryID(self, galleryID):
        self.__galleryID = galleryID

    def set_name(self, name):
        self.__name = name

    def set_website(self, website):
        self.__website = website

    def set_description(self, description):
        self.__description = description

    def set_location(self, location):
        self.__location = location

    def set_curator(self, curator):
        self.__curator = curator

    def set_opening_hours(self, opening_hours):
        self.__opening_hours = opening_hours
```

5.User Favorite Artwork

```
class UserFavoriteArtwork:
    def __init__(self, userID, artworkID):
        self.__userID = userID
        self.__artworkID = artworkID

    def get_user_id(self):
        return self.__userID

    def set_user_id(self, userID):
        self.__userID = userID

    def get_artwork_id(self):
        return self.__artworkID

    def set_artwork_id(self, artworkID):
        self.__artworkID = artworkID
```

6.Service Provider Interface/Abstract class Keep the interfaces and implementation classes in package dao Create IVirtualArtGallery Interface/abstract class with the following methods

```
from abc import ABC, abstractmethod
from entity.Artwork import Artwork
from entity.user import User
from typing import List

class IVirtualArtGallery(ABC):
    @abstractmethod
    def add_artwork(self, artwork: Artwork) -> bool:
        pass

    @abstractmethod
    def update_artwork(self, artwork: Artwork) -> bool:
        pass

    @abstractmethod
    def remove_artwork(self, artworkId: int) -> bool:
        pass

    @abstractmethod
    def get_artwork_by_id(self, artworkId: int) -> Artwork:
        pass

    @abstractmethod
    def search_artworks(self, keyword: str) -> List[Artwork]:
        pass

    @abstractmethod
    def add_artwork_to_favorite(self, userID: int, artworkId: int) -> bool:
        pass

    @abstractmethod
    def remove_artwork_from_favorites(self, userID: int, artworkId: int) -> bool:
        pass

    @abstractmethod
    def get_user_favorite_artworks(self, userID: int) -> List[Artwork]:
        pass

    @abstractmethod
    def create_new_gallery(self, gallery):
        pass

    @abstractmethod
    def update_gallery(self, gallery):
        pass

    @abstractmethod
    def get_gallery_by_id(self,gallery):
        pass

    @abstractmethod
    def remove_gallery(self, gallery):
        pass

    @abstractmethod
    def search_gallery(self, keyword):
        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 fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

```
import mysql.connector
from util.PropertyUtil import PropertyUtil
class DBConnection:
    connection=None
    @staticmethod
    def getConnection():
        # property=PropertyUtil.getPropertyString()
        if DBConnection.connection is None:
            connecting_string=PropertyUtil.getPropertyString()
            try:
                DBConnection.connection=mysql.connector.connect(

                        host=connecting_string["host"],
                        user=connecting_string["user"],
                        password=connecting_string["password"],
                        database=connecting_string["database"],
                        port=connecting_string["port"],


                )
            except mysql.connector.Error as error:
                print(f"Error occurs: {error}")
        return DBConnection.connection


        return conn

class PropertyUtil:
    @staticmethod
    def getPropertyString():
        return{
            "host":"localhost",
            "database":"virtualart_gallery",
            "user":"root",
            "password":"diana",
            "port":"3306",



        }
```

8: Service implementation

. Create a Service class CrimeAnalysisServiceImpl in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the getConnection() method in DBConnection class 2. Provide implementation for all the methods in the interface.

// 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 }

```
from typing import List
from entity.Artwork import Artwork
from entity.gallery import Gallery
from util.DBconnection import DBConnection
```

```
from dao.IVirtualArtgallery import IVirtualArtGallery
from exceptionpack.ArtworkNotFoundException import ArtworkNotFoundException
from exceptionpack.UserNotFoundException import UserNotFoundException

class serviceprovider(IVirtualArtGallery):
    def __init__(self):
        self.connection = DBConnection.getConnection()
        self.cursor = self.connection.cursor()

    def add_artwork(self, artwork: Artwork) -> bool:
        query = (
            "INSERT INTO artwork (ArtworkID, Title, Description, CreationDate, Medium,
ImageURL) "
            "VALUES (%s, %s, %s, %s, %s, %s)"
        )
        values = (
            artwork.get_artworkId(), artwork.get_title(), artwork.get_description(),
artwork.get_creationDate(), artwork.get_medium(), artwork.get_imageUrl()
        )
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    def update_artwork(self, artwork: Artwork) -> bool:
        query = (
            "UPDATE artwork SET Title = %s, Description = %s, CreationDate = %s, Medium
= %s, ImageURL = %s "
            "WHERE ArtworkID = %s"
        )
        values = (
            artwork.get_title(), artwork.get_description(), artwork.get_creationDate(),
artwork.get_medium(), artwork.get_imageUrl(), artwork.get_artworkId()
        )
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    def remove_artwork(self, artwork_id: int) -> bool:
        query = "DELETE FROM artwork WHERE ArtworkID = %s"
        self.cursor.execute(query, (artwork_id,))
        self.connection.commit()
        return True

    def get_artwork_by_id(self, artwork_id: int) -> Artwork:
        query = "SELECT * FROM artwork WHERE ArtworkID = %s"
        self.cursor.execute(query, (artwork_id,))
        result = self.cursor.fetchone()
        if result:
            artwork = Artwork(
                artworkId=result[0],
                title=result[1],
                description=result[2],
                creationDate=result[3],
                medium=result[4],
                imageUrl=result[5]
            )
            return artwork
        else:
            raise ArtworkNotFoundException(artwork_id)

    def search_artworks(self, keyword: str) -> List[Artwork]:
        query = "SELECT * FROM artwork WHERE Title LIKE %s OR Description LIKE %s"
        self.cursor.execute(query, ("%" + keyword + "%", "%" + keyword + "%"))
        results = self.cursor.fetchall()
        artworks = []
        for result in results:
            artwork = Artwork(
                artworkId=result[0],
                title=result[1],
```

```
                description=result[2],
                creationDate=result[3],
                medium=result[4],
                imageUrl=result[5]
            )
            artworks.append(artwork)
        return artworks

    def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
        query = "INSERT INTO user_favorite_artwork (UserID, ArtworkID) VALUES (%s, %s)"
        values = (user_id, artwork_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    def remove_artwork_from_favorites(self, user_id: int, artwork_id: int) -> bool:
        query = "DELETE FROM user_favorite_artwork WHERE UserID = %s AND ArtworkID =
%s"
        values = (user_id, artwork_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    def get_user_favorite_artworks(self, user_id: int) -> List[Artwork]:
        query = "SELECT * FROM artwork WHERE ArtworkID IN (SELECT ArtworkID FROM
user_favorite_artwork WHERE UserID = %s)"
        self.cursor.execute(query, (user_id,))
        results = self.cursor.fetchall()
        artworks = []
        for result in results:
            artwork = Artwork(
                artworkId=result[0],
                title=result[1],
                description=result[2],
                creationDate=result[3],
                medium=result[4],
                imageUrl=result[5]
            )
            artworks.append(artwork)
        return artworks

    def create_new_gallery(self, gallery: Gallery) -> bool:
        query = (
            "INSERT INTO gallery (GalleryID, Name, Website, Description, Location,
Curator, OpeningHours) "
            "VALUES (%s, %s, %s, %s, %s, %s, %s)"
        )
        values = (
            gallery.get_galleryID(),
            gallery.get_name(),
            gallery.get_website(),
            gallery.get_description(),
            gallery.get_location(),
            gallery.get_curator(),
            gallery.get_opening_hours(),
        )
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    def get_gallery_by_id(self, gallery_id: int) -> Gallery:
        query = "SELECT * FROM gallery WHERE GalleryID = %s"
        self.cursor.execute(query, (gallery_id,))
        result = self.cursor.fetchone()
        if result:
            return Gallery(
                galleryID=result[0],
                name=result[1],
                website=result[2],
```

```
                description=result[3],
                location=result[4],
                curator=result[5],
                opening_hours=result[6]
            )
        else:
            return None

    def update_gallery(self, gallery: Gallery) -> bool:
        query = (
            "UPDATE gallery SET Name = %s, Website= %s, Description = %s, Location =
%s, Curator = %s, OpeningHours = %s "
            "WHERE GalleryID = %s"
        )
        values = (
            gallery.get_name(),
            gallery.get_website(),
            gallery.get_description(),
            gallery.get_location(),
            gallery.get_curator(),
            gallery.get_opening_hours(),
            gallery.get_galleryID()
        )
        self.cursor.execute(query, values)
        self.connection.commit()
        return True
    def remove_gallery(self, gallery_id: int) -> bool:
        query = "DELETE FROM gallery WHERE GalleryID = %s"
        self.cursor.execute(query, (gallery_id,))
        self.connection.commit()
        return True

    def search_gallery(self, keyword: str) -> List[Gallery]:
        query = "SELECT * FROM gallery WHERE Name LIKE %s OR Description LIKE %s OR
Location LIKE %s"
        self.cursor.execute(query, ("%" + keyword + "%", "%" + keyword + "%", "%" +
keyword + "%"))
        results = self.cursor.fetchall()
        galleries = []
        for result in results:
            gallery = Gallery(
                galleryID=result[0],
                name=result[1],
                website=result[2],
                description=result[3],
                location=result[4],
                curator=result[5],
                opening_hours=result[6]
            )
            galleries.append(gallery)
        return galleries
```

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

```
class ArtworkNotFoundException(Exception):
    def __init__(self,artworkId):
        self.artworkId=artworkId
        super().__init__(f"Artwork with ID{artworkId} not found in the Artwork")
```

```
Enter your choice: 4
Enter artwork ID: 17
Artwork with ID17 not found in the Artwork
```

```
class UserNotFoundException(Exception):
    def __init__(self,userID):
        self.userID = userID
        super().__init__(f"Artwork with ID{userID} not found in the User")
```

```
Enter your choice: 8
Enter user ID: 17
User Favorite not found.
```

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

```
from entity.Artwork import Artwork
from entity.gallery import Gallery
from exceptionpack.ArtworkNotFoundException import  ArtworkNotFoundException
from dao.IVirtualArtgallery import IVirtualArtGallery
from dao.serviceprovider import serviceprovider
class MainModule:
    def __init__(self):
        self.service = serviceprovider()

    def main(self):
        while True:
            print("----------------------------")
            print()
            print("1. Add Artwork")
            print("2. Update Artwork")
            print("3. Remove Artwork")
            print("4. Get Artwork by ID")
            print("5. Search Artworks")
            print("6. Add Artwork to Favorites")
            print("7. Remove Artwork from Favorites")
            print("8. Get User's Favorite Artworks")
            print("9. create new Gallery")
            print("10. update gallery ")
            print("11. remove gallery")
            print("12. search gallery")
            print("13. Exit")
            print()
            print("-------------------------------")

            choice = input("Enter your choice: ")

            if choice == "1":
                self.add_artwork()
            elif choice == "2":
                self.update_artwork()
            elif choice == "3":
                self.remove_artwork()
            elif choice == "4":
                self.get_artwork_by_id()
            elif choice == "5":
                self.search_artworks()
```

```
            elif choice == "6":
                self.add_artwork_to_favorites()
            elif choice == "7":
                self.remove_artwork_from_favorites()
            elif choice == "8":
                self.get_user_favorite_artworks()
            elif choice == "9":
                self.create_new_gallery()
            elif choice == "10":
                self.update_gallery()
            elif choice == "11":
                self.remove_gallery()
            elif choice == "12":
                self.search_gallery()
            elif choice=="13":
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please enter a valid option.")


    def add_artwork(self):
        artworkId = int(input("Enter artwork ID: "))
        title = input("Enter title: ")
        description = input("Enter description: ")
        creationDate = input("Enter creation date: ")
        medium = input("Enter medium: ")
        imageUrl = input("Enter image URL: ")
        artwork = Artwork(
            artworkId=artworkId,
            title=title,
            description=description,
            creationDate=creationDate,
            medium=medium,
            imageUrl=imageUrl,
        )
        self.service.add_artwork(artwork)
        print("Artwork added successfully!!!")

    def update_artwork(self):
        artwork_id = int(input("Enter artwork ID: "))
        artwork = self.service.get_artwork_by_id(artwork_id)
        if artwork:
            title = input("Enter the title : ")
            description = input("Enter the description : ")
            creation_date = input("Enter the creation date: ")
            medium = input("Enter the medium : ")
            image_url = input("Enter the image URL: ")
            if title:
                artwork.Title = title
            if description:
                artwork.Description = description
            if creation_date:
                artwork.CreationDate = creation_date
            if medium:
                artwork.Medium = medium
            if image_url:
                artwork.ImageUrl = image_url
            self.service.update_artwork(artwork)
            print("Artwork updated successfully!!!")
        else:
            print("Artwork not found.")

    def remove_artwork(self):
        artwork_id = int(input("Enter artwork ID: "))
        try:
            self.service.remove_artwork(artwork_id)
            print("Artwork removed successfully.")
        except ArtworkNotFoundException:
```

```
            print("Artwork not found.")

    def get_artwork_by_id(self):
        artwork_id = int(input("Enter artwork ID: "))
        try:
            artwork = self.service.get_artwork_by_id(artwork_id)
            if artwork:
                print("Artwork details:")
                print(artwork)
            else:
                print("Artwork not found.")
        except ArtworkNotFoundException as e:
            print(e)

    def search_artworks(self):
        keyword = input("Enter keyword to search: ")
        artworks = self.service.search_artworks(keyword)
        if artworks:
            print("Artworks Matching:")
            for artwork in artworks:
                print(artwork)
        else:
            print("No artworks found.")

    def add_artwork_to_favorites(self):
        user_id = int(input("Enter user ID: "))
        artwork_id = int(input("Enter artwork ID: "))
        self.service.add_artwork_to_favorite(user_id, artwork_id)
        print("Artwork added to favorites.")

    def remove_artwork_from_favorites(self):
        user_id = int(input("Enter user ID: "))
        artwork_id = int(input("Enter artwork ID: "))
        self.service.remove_artwork_from_favorites(user_id, artwork_id)
        print("Artwork removed from favorites.")

    def get_user_favorite_artworks(self):
        user_id = int(input("Enter user ID: "))
        artwork = self.service.get_user_favorite_artworks(user_id)
        if artwork:
            print("User Favorite details:")
            for i in artwork:
                print(i)
        else:
            print("User Favorite not found.")


    def create_new_gallery(self):
        gallery_id = int(input("Enter gallery ID: "))
        name = input("Enter gallery name: ")
        website = input("Enter the website: ")
        description = input("Enter gallery description: ")
        location = input("Enter gallery location: ")
        curator = int(input("Enter curator (artist ID): "))
        opening_hours = input("Enter opening hours: ")

        gallery = Gallery(
            galleryID=gallery_id,
            name=name,
            website=website,
            description=description,
            location=location,
            curator=curator,
            opening_hours=opening_hours,
        )
        self.service.create_new_gallery(gallery)
        print("Gallery created successfully.")

    def update_gallery(self):
```

```
        gallery_id = int(input("Enter galleryid: "))
        gallery = self.service.get_gallery_by_id(gallery_id)
        if gallery:
            name = input("Enter new name : ")
            description = input("Enter new description:  ")
            location = input("Enter new location : ")
            curator = input("Enter new curator : ")
            openinghours = input("Enter new openinghours:   ")
            if name:
                gallery.Name = name
            if description:
                gallery.Description = description
            if location:
                gallery.Location = location
            if curator:
                gallery.Curator = curator
            if openinghours:
                gallery.OpeningHours = openinghours
            self.service.update_gallery(gallery)
            print("Gallery updated successfully.")
        else:
            print("Gallery not found.")



    def remove_gallery(self):
        gallery_id = int(input("Enter gallery ID: "))
        self.service.remove_gallery(gallery_id)
        print("Gallery removed successfully.")

    def search_gallery(self):
        keyword = input("Enter keyword to search: ")
        galleries = self.service.search_gallery(keyword)
        if galleries:
            print("Matching artworks:")
            for gallery in galleries:
                print(gallery)
        else:
            print("No matching artworks found.")



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

```
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User's Favorite Artworks
9. create new Gallery
10. update gallery
11. remove gallery
12. search gallery
13. Exit

--------------------------------
Enter your choice: 1
Enter artwork ID: 14
Enter title: humid
Enter description: cool
Enter creation date: 2024-03-10
Enter medium: coolday
Enter image URL: www.cool.com
Artwork added successfully!!!
```

2	Mona Lisa	A renowned portrait painting by Leonardo da Vi...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Starry Night	A painting by Edvard Munch depicting a screami...	1893-01-01	Oil, tempera, and pastel on cardboard	https://example.com/scream.jpg
5	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
6	Girl with a Pearl Earring	A portrait painting by Johannes Vermeer.	1665-01-01	Oil on canvas	https://example.com/girl_with_pearl_earring.jpg
7	The Birth of Venus	A painting by Sandro Botticelli depicting the god...	1486-01-01	Tempera on canvas	https://example.com/birth_of_venus.jpg
8	The Great Wave off Kanagawa	A woodblock print by Katsushika Hokusai.	1831-01-01	Woodblock print	https://example.com/great_wave.jpg
9	Guernica	A mural-sized painting by Pablo Picasso depictin...	1937-01-01	Oil on canvas	https://example.com/guernica.jpg
10	Water Lilies	A series of approximately 250 oil paintings by Cl...	1916-01-01	Oil on canvas	https://example.com/water_lilies.jpg
13	dark of horror	horror	2024-01-01	horrorflim	www.flim.com
14	humid	cool	2024-03-10	coolday	www.cool.com

```
Enter your choice: 4
Enter artwork ID: 3
Artwork with ID3 not found in the Artwork
----------------------------
```

2	Mona Lisa	A renowned portrait painting by Leonardo da Vi...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Starry Night	A painting by Edvard Munch depicting a screami...	1893-01-01	Oil, tempera, and pastel on cardboard	https://example.com/scream.jpg
5	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
6	Girl with a Pearl Earring	A portrait painting by Johannes Vermeer.	1665-01-01	Oil on canvas	https://example.com/girl_with_pearl_earring.jpg
7	The Birth of Venus	A painting by Sandro Botticelli depicting the god...	1486-01-01	Tempera on canvas	https://example.com/birth_of_venus.jpg
8	The Great Wave off Kanagawa	A woodblock print by Katsushika Hokusai.	1831-01-01	Woodblock print	https://example.com/great_wave.jpg

```
Enter your choice: 6
Enter user ID: 1
Enter artwork ID: 6
Artwork added to favorites.
----------------------------
```

UserID	ArtworkID
1	1
1	3
1	6
2	2
2	4
2	5
3	3
3	5
4	4

```
Enter your choice: 4
Enter artwork ID: 4
Artwork details:
Artwork ID: 4, Title: The Starry Night, Description: A painting by Edvard Munch depicting a screaming figure against a colorful sky., Creation Date: 1893-01-01, Medium: Oil,
----------------------------
```

ArtworkID	Title	Description	CreationDate	Medium	ImageURL
2	Mona Lisa	A renowned portrait painting by Leonardo da Vi...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Starry Night	A painting by Edvard Munch depicting a screami...	1893-01-01	Oil, tempera, and pastel on cardboard	https://example.com/scream.jpg
5	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
6	Girl with a Pearl Earring	A portrait painting by Johannes Vermeer.	1665-01-01	Oil on canvas	https://example.com/girl_with_pearl_earring.jpg
7	The Birth of Venus	A painting by Sandro Botticelli depicting the god...	1486-01-01	Tempera on canvas	https://example.com/birth_of_venus.jpg
8	The Great Wave off Kanagawa	A woodblock print by Katsushika Hokusai.	1831-01-01	Woodblock print	https://example.com/great_wave.jpg
9	Guernica	A mural-sized painting by Pablo Picasso depictin...	1937-01-01	Oil on canvas	https://example.com/guernica.jpg

```
Enter your choice: 5
Enter keyword to search: Starry
Artworks Matching:
Artwork ID: 4, Title: The Starry Night, Description: A painting by Edvard Munch depicting a screaming figure against a colorful sky., Creation Date: 1893-01-01, Medium: Oil,
----------------------------
```

ArtworkID	Title	Description	CreationDate	Medium	ImageURL
2	Mona Lisa	A renowned portrait painting by Leonardo da Vi...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Starry Night	A painting by Edvard Munch depicting a screami...	1893-01-01	Oil, tempera, and pastel on cardboard	https://example.com/scream.jpg
5	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
6	Girl with a Pearl Earring	A portrait painting by Johannes Vermeer.	1665-01-01	Oil on canvas	https://example.com/girl_with_pearl_earring.jpg
7	The Birth of Venus	A painting by Sandro Botticelli depicting the god...	1486-01-01	Tempera on canvas	https://example.com/birth_of_venus.jpg
8	The Great Wave off Kanagawa	A woodblock print by Katsushika Hokusai.	1831-01-01	Woodblock print	https://example.com/great_wave.jpg
9	Guernica	A mural-sized painting by Pablo Picasso depictin...	1937-01-01	Oil on canvas	https://example.com/guernica.jpg
10	Water Lilies	A series of approximately 250 oil paintings by Cl...	1916-01-01	Oil on canvas	https://example.com/water_lilies.jpg

```
Enter your choice: 6
Enter user ID: 14
Enter artwork ID: 12
Artwork added to favorites.
```

UserID	ArtworkID
3	3
3	5
4	4
4	6
5	5
5	7
7	2
14	12
NULL	NULL

```
Enter your choice: 7
Enter user ID: 14
Enter artwork ID: 12
Artwork removed from favorites.
---------------------------
```

UserID	ArtworkID
2	5
3	3
3	5
4	4
4	6
5	5
5	7
7	2
NULL	NULL

```
--------------------------------
Enter your choice: 9
Enter gallery ID: 15
Enter gallery name: gallery15
Enter the website: www.gallery15.com
Enter gallery description: newgallery
Enter gallery location: location15
Enter curator (artist ID): 11
Enter opening hours: Mon-Fri: 9am-5pm
Gallery created successfully.
```

GalleryID	Name	Website	Description	Location	Curator	OpeningHours
2	Gallery B	https://www.galleryb.com	Specializes in modern and abstract art exhibitions.	Los Angeles	2	Tue-Sat: 11am-7pm
3	Gallery C	https://www.galleryc.com	Showcases traditional and experimental art forms.	London	3	Wed-Sun: 12pm-8pm
4	Gallery D	https://www.galleryd.com	Focuses on photography and digital art.	Paris	4	Mon-Sat: 10am-5pm
5	Gallery E	https://www.gallerye.com	Promotes emerging artists and hosts workshops.	Berlin	5	Thu-Sun: 1pm-9pm
6	dfg	NULL	fgh	fg	6	Fri-Tue: 1pm-9pm
8	jyhsr	www.gallery.com	fhjb	gnn 6	8	Fri-Tue: 1pm-9pm
11	gallery11	www.gallery11.com	newgallery	location11	11	Fri-Tue: 1pm-9pm
15	gallery15	www.gallery15.com	newgallery	location15	11	Mon-Fri: 9am-5pm
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Enter your choice: 11
Enter gallery ID: 8
Gallery removed successfully.
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

GalleryID	Name	Website	Description	Location	Curator	OpeningHours
2	Gallery B	https://www.galleryb.com	Specializes in modern and abstract art exhibitions.	Los Angeles	2	Tue-Sat: 11am-7pm
3	Gallery C	https://www.galleryc.com	Showcases traditional and experimental art forms.	London	3	Wed-Sun: 12pm-8pm
4	Gallery D	https://www.galleryd.com	Focuses on photography and digital art.	Paris	4	Mon-Sat: 10am-5pm
5	Gallery E	https://www.gallerye.com	Promotes emerging artists and hosts workshops.	Berlin	5	Thu-Sun: 1pm-9pm
6	dfg	NULL	fgh	fg	6	Fri-Tue: 1pm-9pm
11	gallery11	www.gallery11.com	newgallery	location11	11	Fri-Tue: 1pm-9pm
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Enter your choice: 12
Enter keyword to search: Showcases
Matching artworks:
Gallery ID: 3, Name: Gallery C, Website: https://www.galleryc.com, Description: Showcases traditional and experimental art forms., Location: London, Curator: 3, Opening Hours:
--------------------------
```

GalleryID	Name	Website	Description	Location	Curator	OpeningHours
2	Gallery B	https://www.galleryb.com	Specializes in modern and abstract art exhibitions.	Los Angeles	2	Tue-Sat: 11am-7pm
3	Gallery C	https://www.galleryc.com	Showcases traditional and experimental art forms.	London	3	Wed-Sun: 12pm-8pm
4	Gallery D	https://www.galleryd.com	Focuses on photography and digital art.	Paris	4	Mon-Sat: 10am-5pm
5	Gallery E	https://www.gallerye.com	Promotes emerging artists and hosts workshops.	Berlin	5	Thu-Sun: 1pm-9pm
6	dfg	NULL	fgh	fg	6	Fri-Tue: 1pm-9pm
11	gallery11	www.gallery11.com	newgallery	location11	11	Fri-Tue: 1pm-9pm
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Enter your choice: 13
Exiting...
```

11. 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 unittest
from dao.IVirtualArtgallery import IVirtualArtGallery
from entity.gallery import Gallery
from entity.Artwork import Artwork
from dao.serviceprovider import serviceprovider


class TestVirtualArtGallery(unittest.TestCase):
    def setUp(self):
        self.service = serviceprovider()

    def test_upload_new_artwork(self):
        artwork = Artwork(
            artworkId=90,
            title="Sample Title",
            description="Sample Description",
            creationDate="2024-05-06",
            medium="Sample Medium",
            imageUrl="http://example.com/image.jpg"
        )
        result = self.service.add_artwork(artwork)
        self.assertTrue(result)

    def test_update_artwork_details(self):
        artwork = Artwork(
            artworkId=1,
            title="Updated Title",
            description="Updated Description",
            creationDate="2024-05-06",
            medium="Updated Medium",
            imageUrl="http://example.com/updated_image.jpg"
        )
        result = self.service.update_artwork(artwork)
        self.assertTrue(result)

    def test_remove_artwork(self):
        result = self.service.remove_artwork(1)
        self.assertTrue(result)

    def test_search_artworks(self):
        keyword = "Starry"
        artworks = self.service.search_artworks(keyword)
        self.assertTrue(len(artworks) > 0)
```

```python
    def test_create_new_gallery(self):
        gallery = Gallery(
            galleryID=1,
            name="Sample Gallery",
            description="Sample Description",
            location="Sample Location",
            curator=1,
            opening_hours="10:00:00"
        )
        result = self.service.create_new_gallery(gallery)
        self.assertTrue(result)

    def test_update_gallery_information(self):
        gallery = Gallery(
            galleryID=1,
            name="Updated Gallery Name",
            description="Updated Description",
            location="Updated Location",
            curator=1,
            opening_hours="11:00:00"
        )
        result = self.service.update_gallery(gallery)
        self.assertTrue(result)

    def test_remove_gallery(self):
        result = self.service.remove_gallery(1)
        self.assertTrue(result)

    def test_search_galleries(self):
        keyword = "Showcases"
        galleries = self.service.search_gallery(keyword)
        self.assertTrue(len(galleries) > 0)


if __name__ == '__main__':
    unittest.main()
```

```
"C:\Users\DIANA INBA MALAR\PycharmProjects\pythonProject2\.venv\Scripts\python.exe" "C:/Program Files/JetBrains/PyCharm Community Edition 2024.1/plugins/python-ce/helpers/py
Testing started at 03:52 pm ...
Launching pytest with arguments C:\Users\DIANA INBA MALAR\PycharmProjects\pythonProject2\testing1\testpy.py --no-header --no-summary -q in C:\Users\DIANA INBA MALAR\PycharmP

============================= test session starts =============================
collecting ... collected 8 items

testpy.py::TestVirtualArtGallery::test_create_new_gallery
testpy.py::TestVirtualArtGallery::test_remove_artwork
testpy.py::TestVirtualArtGallery::test_remove_gallery
testpy.py::TestVirtualArtGallery::test_search_artworks
testpy.py::TestVirtualArtGallery::test_search_galleries
testpy.py::TestVirtualArtGallery::test_update_artwork_details
testpy.py::TestVirtualArtGallery::test_update_gallery_information
testpy.py::TestVirtualArtGallery::test_upload_new_artwork

============================== 8 passed in 0.45s ==============================
PASSED        [ 12%]PASSED           [ 25%]PASSED           [ 37%]PASSED           [ 50%]PASSED          [ 62%]PASSED     [ 75%]PASSED [ 87%]PASSED        [100%]
Process finished with exit code 0
```