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)
- **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

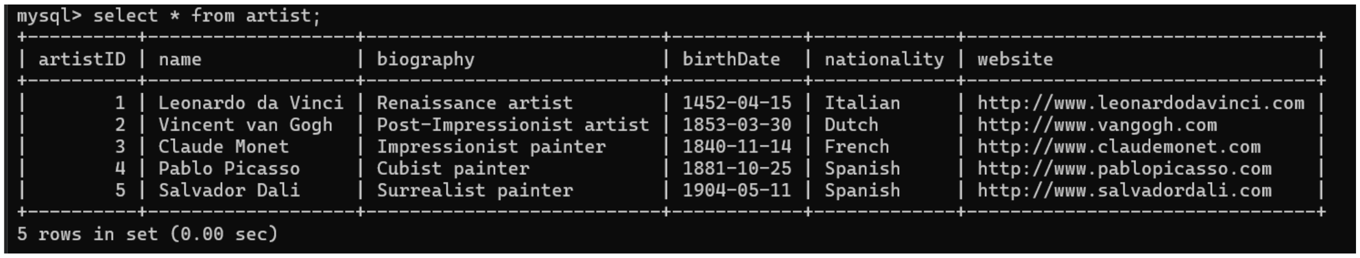
Description

Location

Curator (Reference to ArtistID)

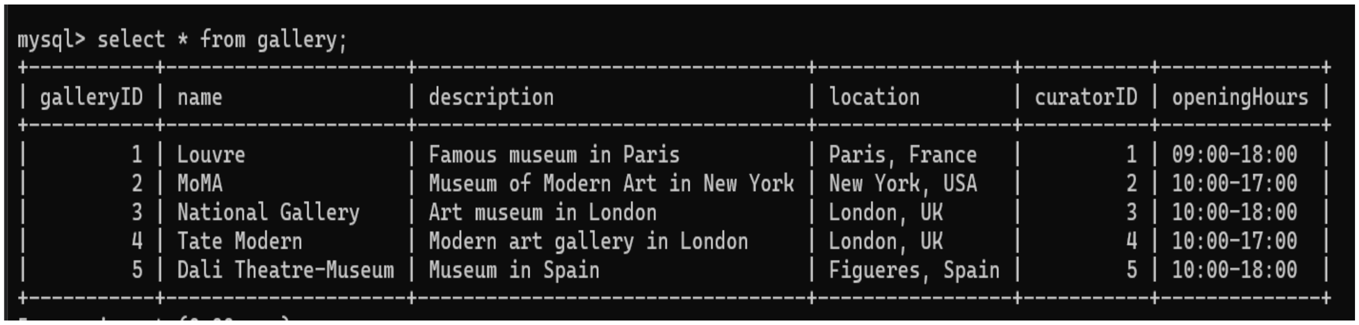
OpeningHours

TABLE Artist



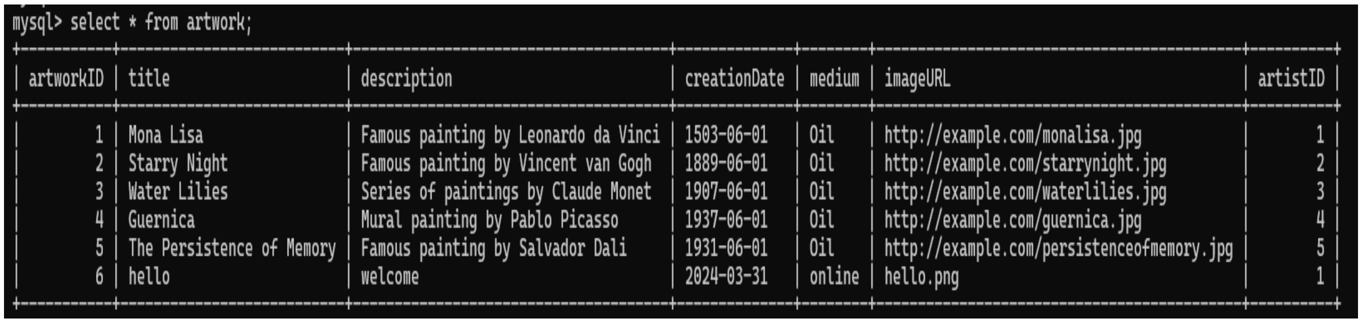
```
mysql> select * from artist;
+----------+-------------------+---------------------------+------------+-------------+---------------------------------+
| artistID | name              | biography                 | birthDate  | nationality | website                         |
+----------+-------------------+---------------------------+------------+-------------+---------------------------------+
|        1 | Leonardo da Vinci | Renaissance artist        | 1452-04-15 | Italian     | http://www.leonardodavinci.com |
|        2 | Vincent van Gogh  | Post-Impressionist artist | 1853-03-30 | Dutch       | http://www.vangogh.com          |
|        3 | Claude Monet      | Impressionist painter     | 1840-11-14 | French      | http://www.claudemonet.com      |
|        4 | Pablo Picasso     | Cubist painter            | 1881-10-25 | Spanish     | http://www.pablopicasso.com     |
|        5 | Salvador Dali     | Surrealist painter        | 1904-05-11 | Spanish     | http://www.salvadordali.com     |
+----------+-------------------+---------------------------+------------+-------------+---------------------------------+
5 rows in set (0.00 sec)
```

TABLE Gallery



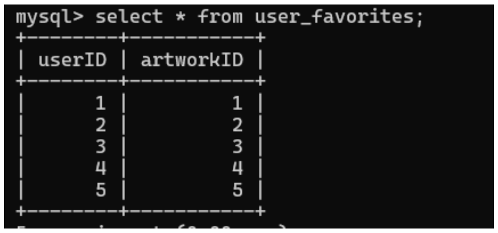
```
mysql> select * from gallery;
+-----------+---------------------+-----------------------------------+-----------------+-----------+--------------+
| galleryID | name                | description                       | location        | curatorID | openingHours |
+-----------+---------------------+-----------------------------------+-----------------+-----------+--------------+
|         1 | Louvre              | Famous museum in Paris            | Paris, France   |         1 | 09:00-18:00  |
|         2 | MoMA                | Museum of Modern Art in New York  | New York, USA   |         2 | 10:00-17:00  |
|         3 | National Gallery    | Art museum in London              | London, UK      |         3 | 10:00-18:00  |
|         4 | Tate Modern         | Modern art gallery in London      | London, UK      |         4 | 10:00-17:00  |
|         5 | Dali Theatre-Museum | Museum in Spain                   | Figueres, Spain |         5 | 10:00-18:00  |
+-----------+---------------------+-----------------------------------+-----------------+-----------+--------------+
```

TABLE Artwork



```
mysql> select * from artwork;
+-----------+---------------------------+-----------------------------------+--------------+--------+-----------------------------------------+----------+
| artworkID | title                     | description                       | creationDate | medium | imageURL                                | artistID |
+-----------+---------------------------+-----------------------------------+--------------+--------+-----------------------------------------+----------+
|         1 | Mona Lisa                 | Famous painting by Leonardo da Vinci | 1503-06-01 | Oil    | http://example.com/monalisa.jpg         |        1 |
|         2 | Starry Night              | Famous painting by Vincent van Gogh  | 1889-06-01 | Oil    | http://example.com/starrynight.jpg      |        2 |
|         3 | Water Lilies              | Series of paintings by Claude Monet  | 1907-06-01 | Oil    | http://example.com/waterlilies.jpg      |        3 |
|         4 | Guernica                  | Mural painting by Pablo Picasso      | 1937-06-01 | Oil    | http://example.com/guernica.jpg         |        4 |
|         5 | The Persistence of Memory | Famous painting by Salvador Dali     | 1931-06-01 | Oil    | http://example.com/persistenceofmemory.jpg |     5 |
|         6 | hello                     | welcome                              | 2024-03-31 | online | hello.png                               |        1 |
+-----------+---------------------------+-----------------------------------+--------------+--------+-----------------------------------------+----------+
```

TABLE User_artwork_favorite:



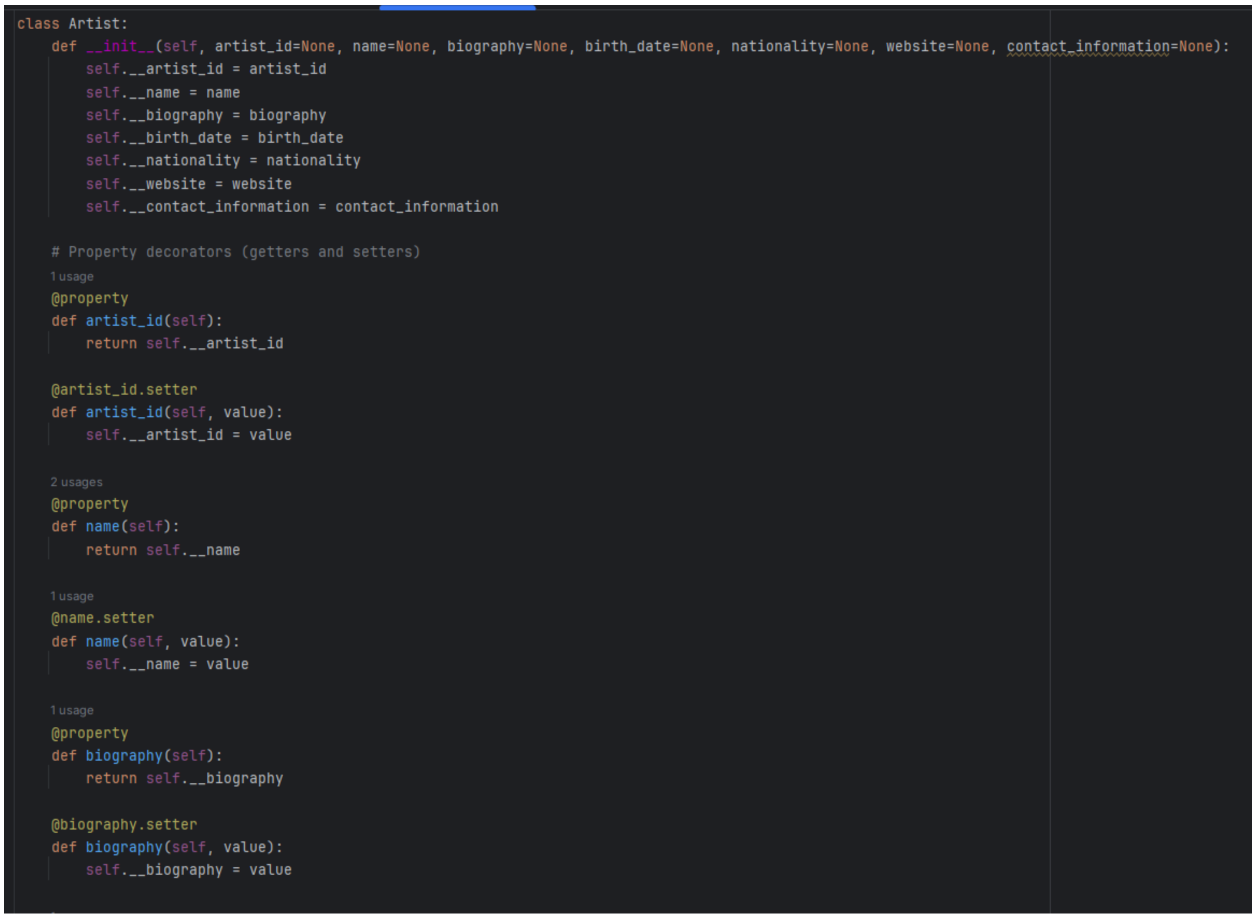
```
mysql> select * from user_favorites;
+--------+-----------+
| userID | artworkID |
+--------+-----------+
|      1 |         1 |
|      2 |         2 |
|      3 |         3 |
|      4 |         4 |
|      5 |         5 |
+--------+-----------+
```

Coding

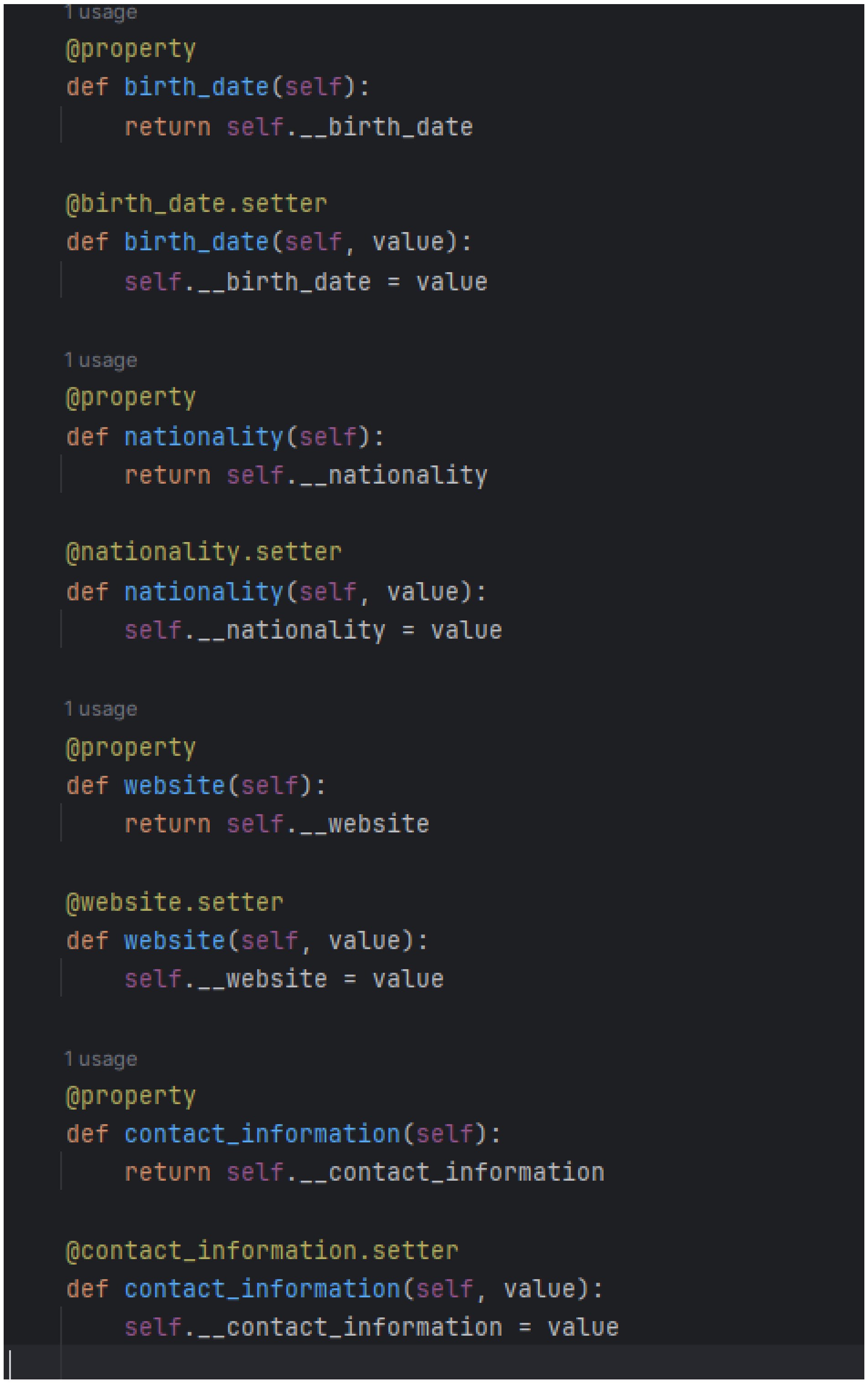
Create the model/entity classes corresponding to the schema **within package** entity **with variables declared private, constructors(default and parametrized) and getters,setters)**

Entity package:

Artist.py

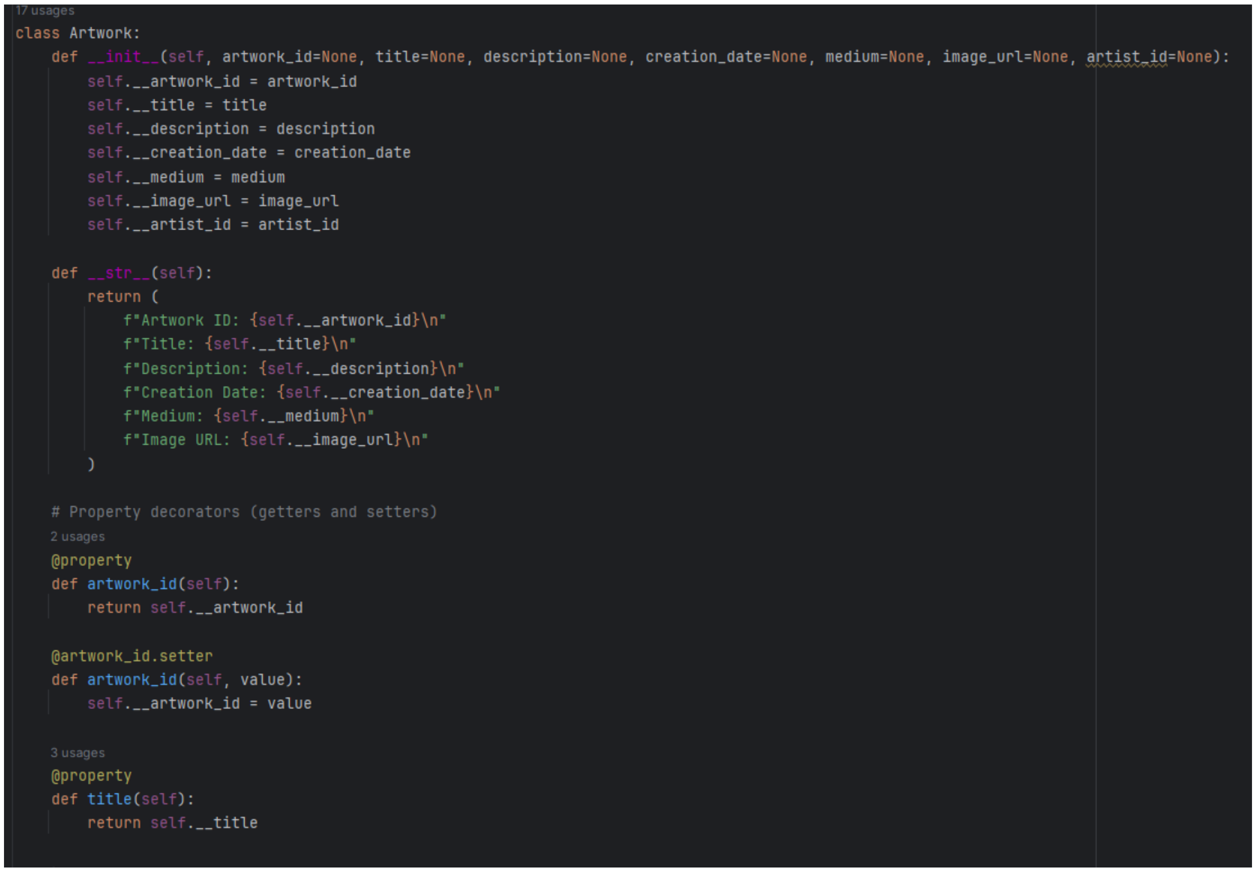


```python
class Artist:
    def __init__(self, artist_id=None, name=None, biography=None, birth_date=None, nationality=None, website=None, contact_information=None):
        self.__artist_id = artist_id
        self.__name = name
        self.__biography = biography
        self.__birth_date = birth_date
        self.__nationality = nationality
        self.__website = website
        self.__contact_information = contact_information

    # Property decorators (getters and setters)
    1 usage
    @property
    def artist_id(self):
        return self.__artist_id

    @artist_id.setter
    def artist_id(self, value):
        self.__artist_id = value

    2 usages
    @property
    def name(self):
        return self.__name

    1 usage
    @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
```

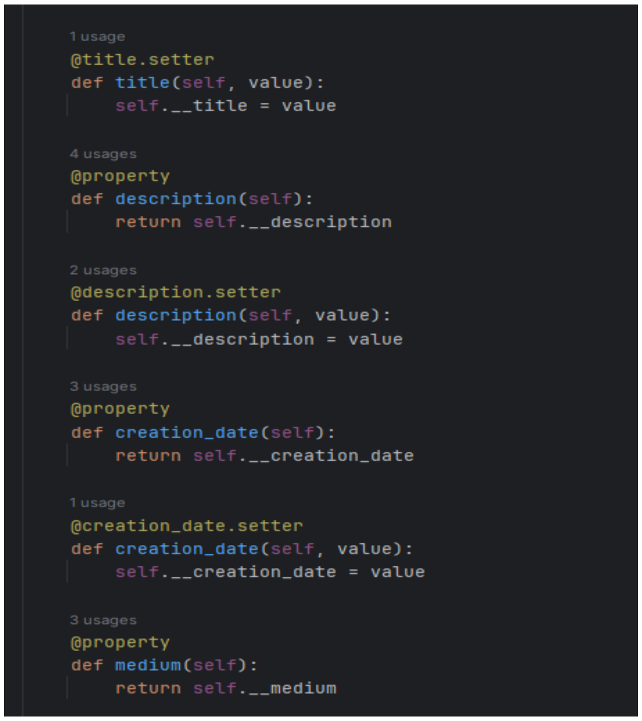


```
    1 usage
    @property
    def birth_date(self):
        return self.__birth_date

    @birth_date.setter
    def birth_date(self, value):
        self.__birth_date = 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 contact_information(self):
        return self.__contact_information

    @contact_information.setter
    def contact_information(self, value):
        self.__contact_information = value
```

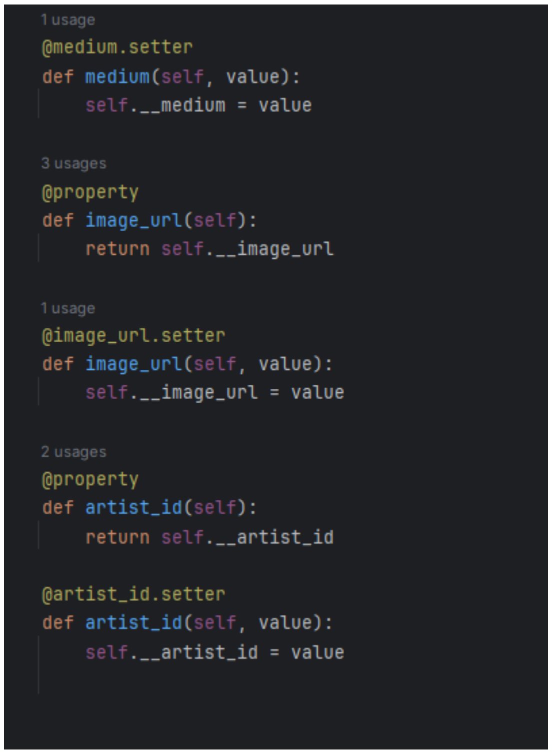
Artwork.py



```
17 usages
class Artwork:
    def __init__(self, artwork_id=None, title=None, description=None, creation_date=None, medium=None, image_url=None, artist_id=None):
        self.__artwork_id = artwork_id
        self.__title = title
        self.__description = description
        self.__creation_date = creation_date
        self.__medium = medium
        self.__image_url = image_url
        self.__artist_id = artist_id

    def __str__(self):
        return (
            f"Artwork ID: {self.__artwork_id}\n"
            f"Title: {self.__title}\n"
            f"Description: {self.__description}\n"
            f"Creation Date: {self.__creation_date}\n"
            f"Medium: {self.__medium}\n"
            f"Image URL: {self.__image_url}\n"
        )

    # Property decorators (getters and setters)
    2 usages
    @property
    def artwork_id(self):
        return self.__artwork_id

    @artwork_id.setter
    def artwork_id(self, value):
        self.__artwork_id = value

    3 usages
    @property
    def title(self):
        return self.__title
```

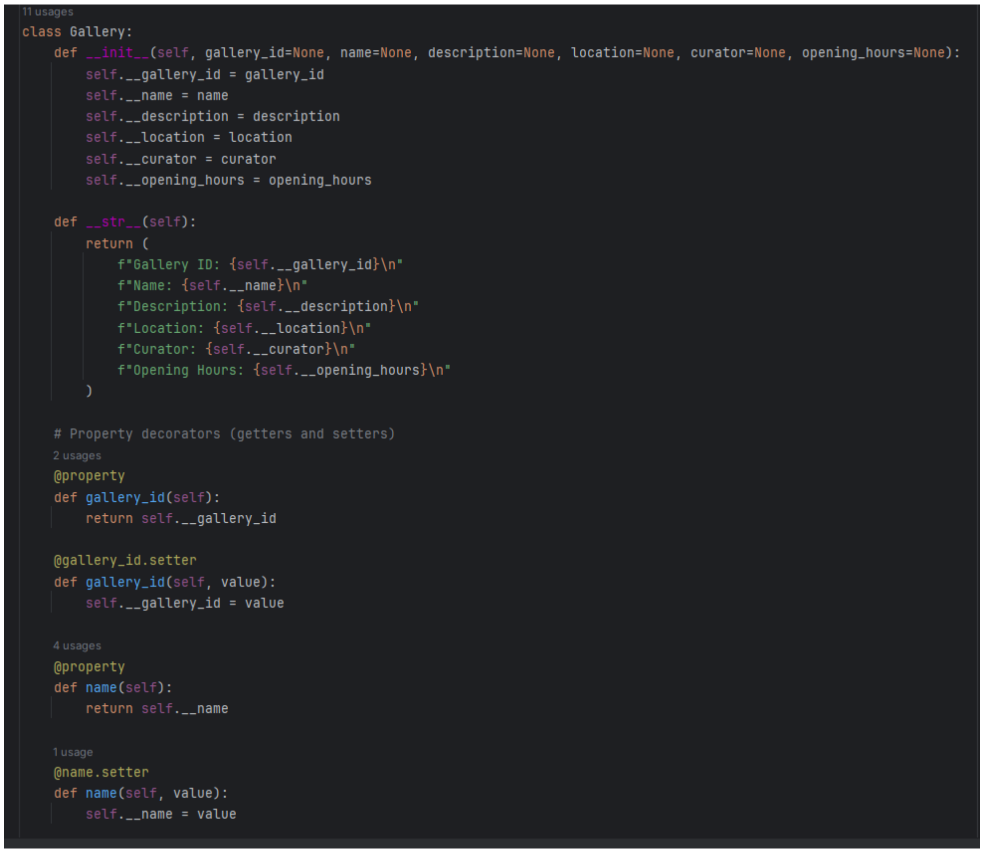


```
    1 usage
    @title.setter
    def title(self, value):
        self.__title = value

    4 usages
    @property
    def description(self):
        return self.__description

    2 usages
    @description.setter
    def description(self, value):
        self.__description = value

    3 usages
    @property
    def creation_date(self):
        return self.__creation_date

    1 usage
    @creation_date.setter
    def creation_date(self, value):
        self.__creation_date = value

    3 usages
    @property
    def medium(self):
        return self.__medium
```

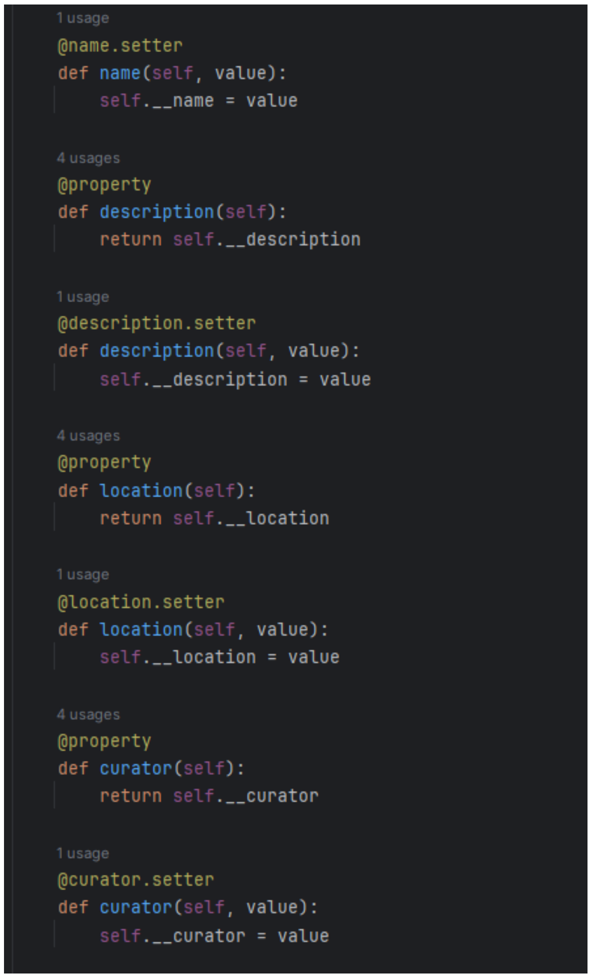


```
1 usage
@medium.setter
def medium(self, value):
    self.__medium = value

3 usages
@property
def image_url(self):
    return self.__image_url

1 usage
@image_url.setter
def image_url(self, value):
    self.__image_url = value

2 usages
@property
def artist_id(self):
    return self.__artist_id

@artist_id.setter
def artist_id(self, value):
    self.__artist_id = value
```

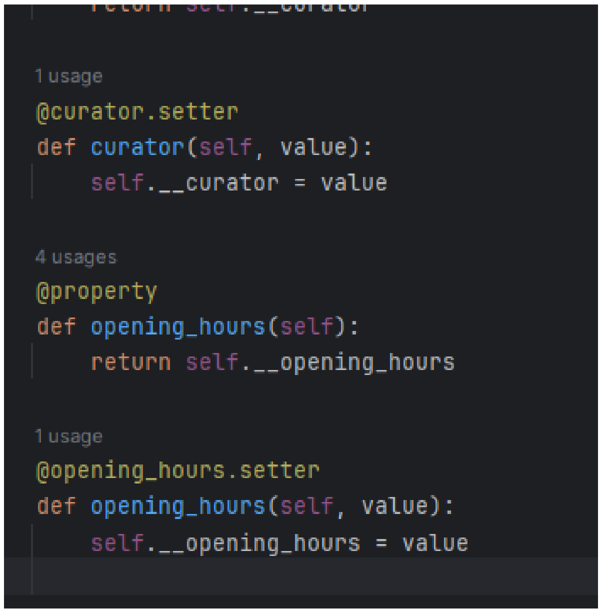
Gallery.py



```
11 usages
class Gallery:
    def __init__(self, gallery_id=None, name=None, description=None, location=None, curator=None, opening_hours=None):
        self.__gallery_id = gallery_id
        self.__name = name
        self.__description = description
        self.__location = location
        self.__curator = curator
        self.__opening_hours = opening_hours

    def __str__(self):
        return (
            f"Gallery ID: {self.__gallery_id}\n"
            f"Name: {self.__name}\n"
            f"Description: {self.__description}\n"
            f"Location: {self.__location}\n"
            f"Curator: {self.__curator}\n"
            f"Opening Hours: {self.__opening_hours}\n"
        )

    # Property decorators (getters and setters)
    2 usages
    @property
    def gallery_id(self):
        return self.__gallery_id

    @gallery_id.setter
    def gallery_id(self, value):
        self.__gallery_id = value

    4 usages
    @property
    def name(self):
        return self.__name

    1 usage
    @name.setter
    def name(self, value):
        self.__name = value
```

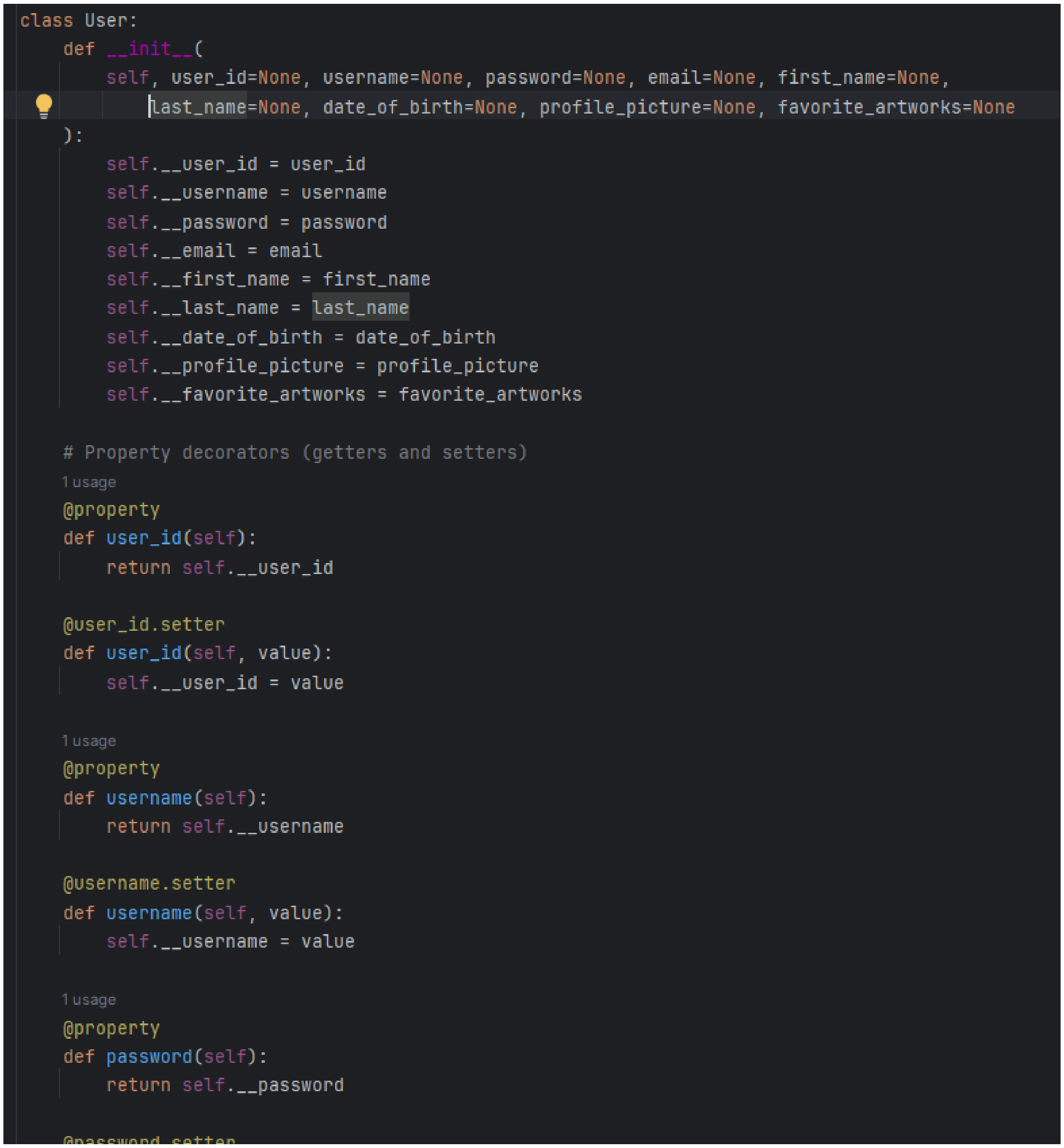


```python
    1 usage
    @name.setter
    def name(self, value):
        self.__name = value

    4 usages
    @property
    def description(self):
        return self.__description

    1 usage
    @description.setter
    def description(self, value):
        self.__description = value

    4 usages
    @property
    def location(self):
        return self.__location

    1 usage
    @location.setter
    def location(self, value):
        self.__location = value

    4 usages
    @property
    def curator(self):
        return self.__curator

    1 usage
    @curator.setter
    def curator(self, value):
        self.__curator = value
```

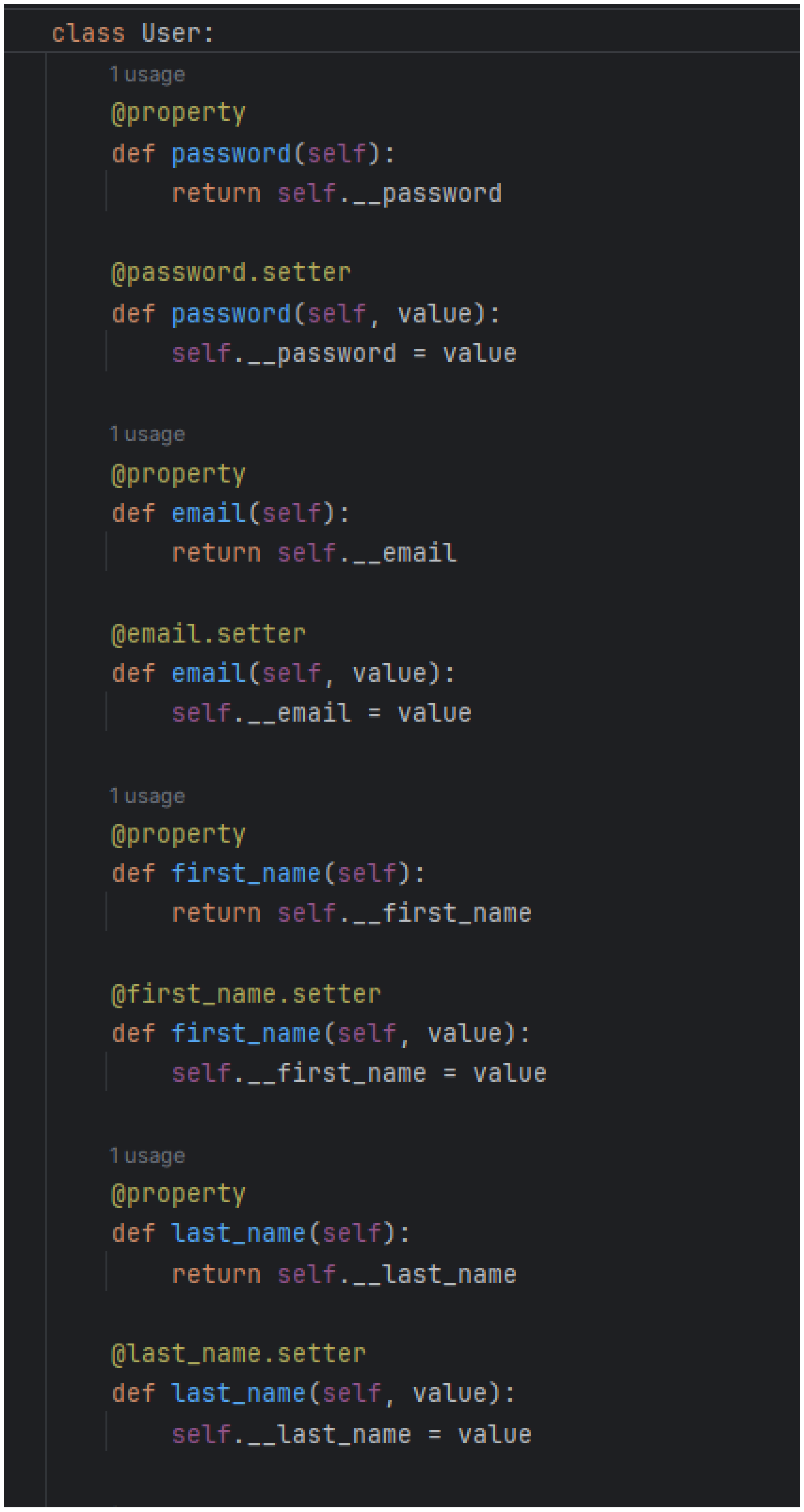


```python
        return self.__curator

    1 usage
    @curator.setter
    def curator(self, value):
        self.__curator = value

    4 usages
    @property
    def opening_hours(self):
        return self.__opening_hours

    1 usage
    @opening_hours.setter
    def opening_hours(self, value):
        self.__opening_hours = value
```

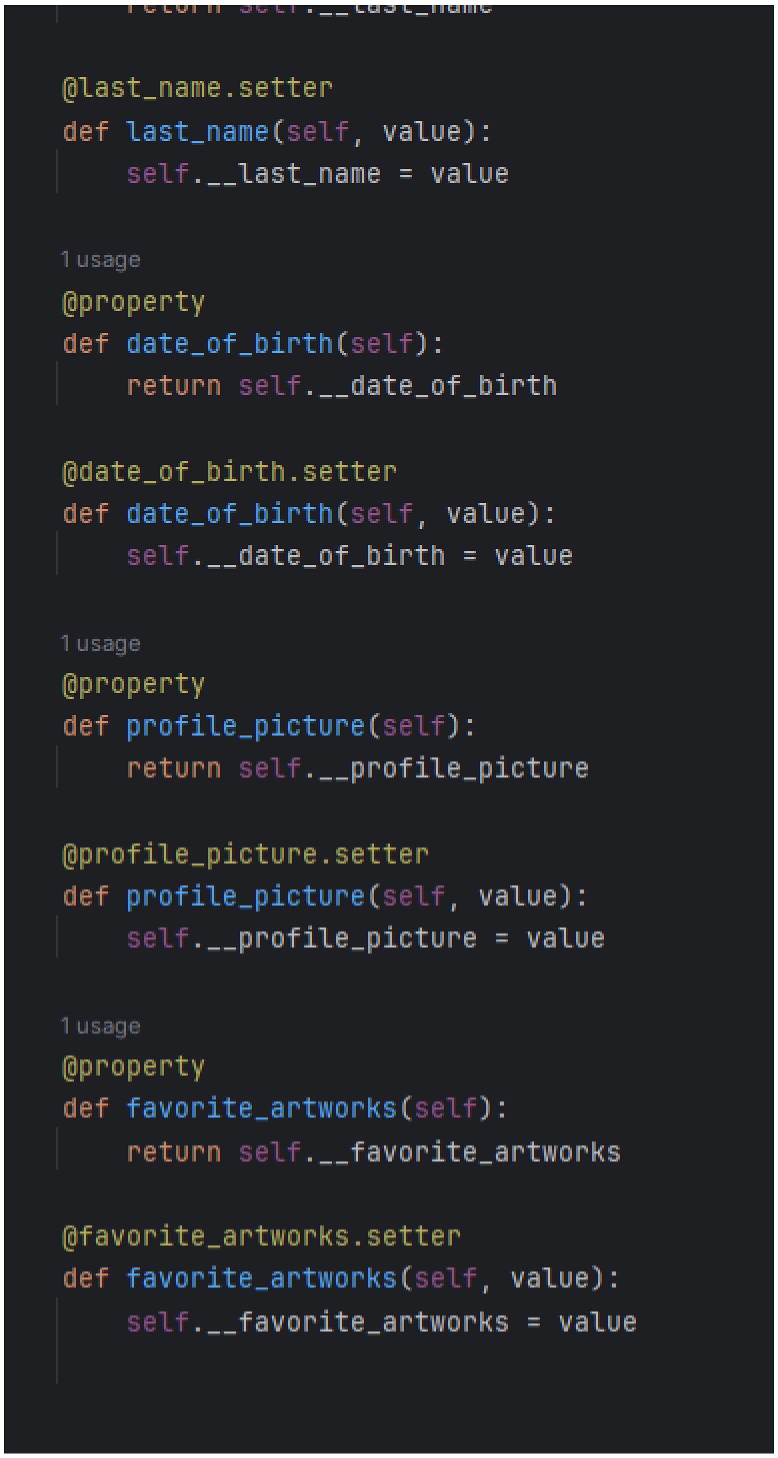
User.py



```python
class User:
    def __init__(
        self, user_id=None, username=None, password=None, email=None, first_name=None,
            last_name=None, date_of_birth=None, profile_picture=None, 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

    # Property decorators (getters and setters)
    1 usage
    @property
    def user_id(self):
        return self.__user_id

    @user_id.setter
    def user_id(self, value):
        self.__user_id = 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
```

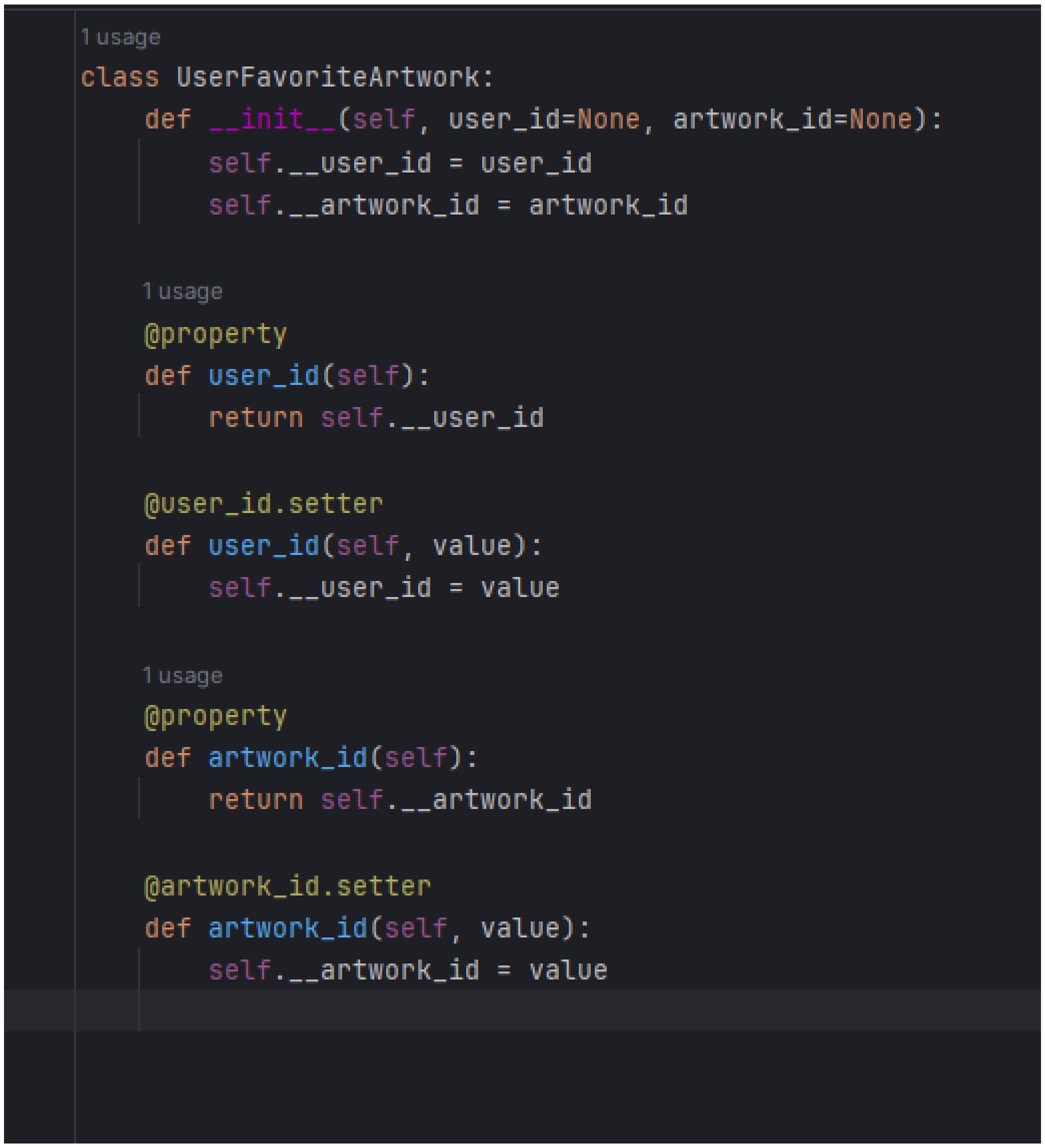


```
class User:
    1 usage
    @property
    def password(self):
        return self.__password

    @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 first_name(self):
        return self.__first_name

    @first_name.setter
    def first_name(self, value):
        self.__first_name = value

    1 usage
    @property
    def last_name(self):
        return self.__last_name

    @last_name.setter
    def last_name(self, value):
        self.__last_name = value
```



```python
    return self.__last_name

@last_name.setter
def last_name(self, value):
    self.__last_name = value

1 usage
@property
def date_of_birth(self):
    return self.__date_of_birth

@date_of_birth.setter
def date_of_birth(self, value):
    self.__date_of_birth = value

1 usage
@property
def profile_picture(self):
    return self.__profile_picture

@profile_picture.setter
def profile_picture(self, value):
    self.__profile_picture = value

1 usage
@property
def favorite_artworks(self):
    return self.__favorite_artworks

@favorite_artworks.setter
def favorite_artworks(self, value):
    self.__favorite_artworks = value
```

UserFavoriteArtwork.py



```
1 usage
class UserFavoriteArtwork:
    def __init__(self, user_id=None, artwork_id=None):
        self.__user_id = user_id
        self.__artwork_id = artwork_id

    1 usage
    @property
    def user_id(self):
        return self.__user_id

    @user_id.setter
    def user_id(self, value):
        self.__user_id = value

    1 usage
    @property
    def artwork_id(self):
        return self.__artwork_id

    @artwork_id.setter
    def artwork_id(self, value):
        self.__artwork_id = 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

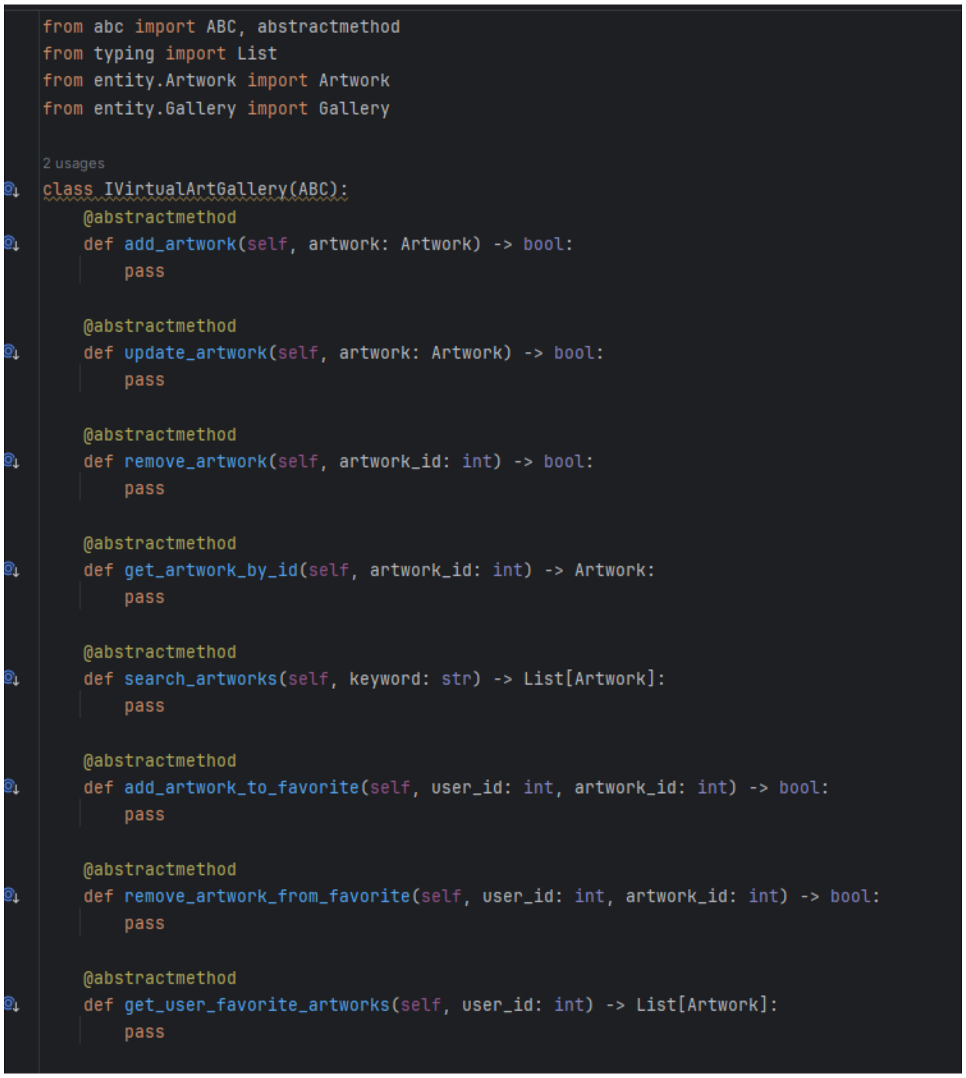
// User Favorites **addArtworkToFavorite**(); parameters- userId, artworkId return type boolean

removeArtworkFromFavorite()
parameters- userId, artworkId
return type boolean

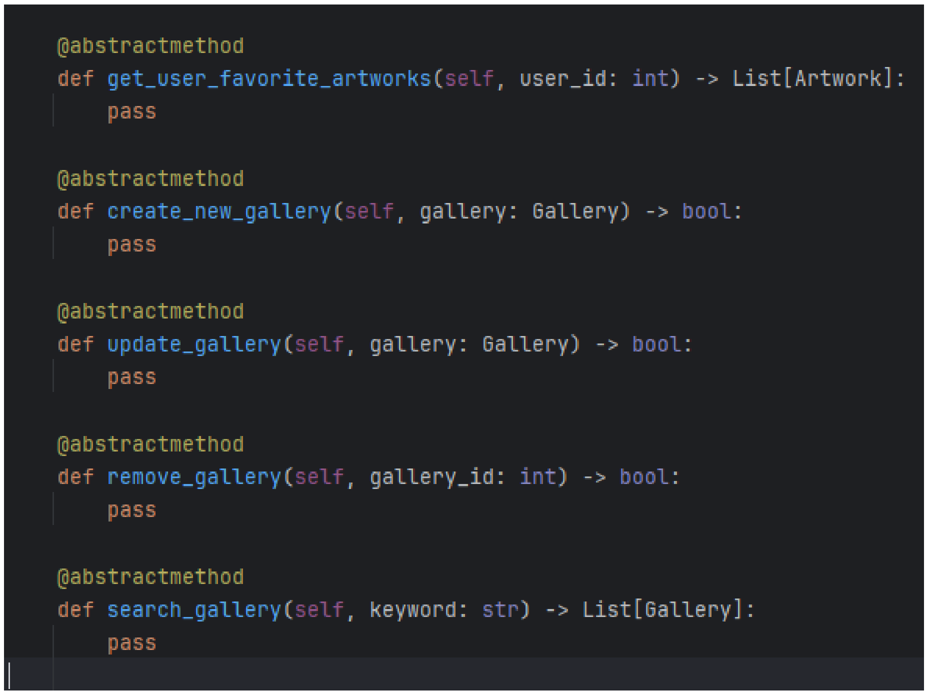
getUserFavoriteArtworks()
parameters- userId
return type boolean

dao package:

IVirtualArtGallery.py:

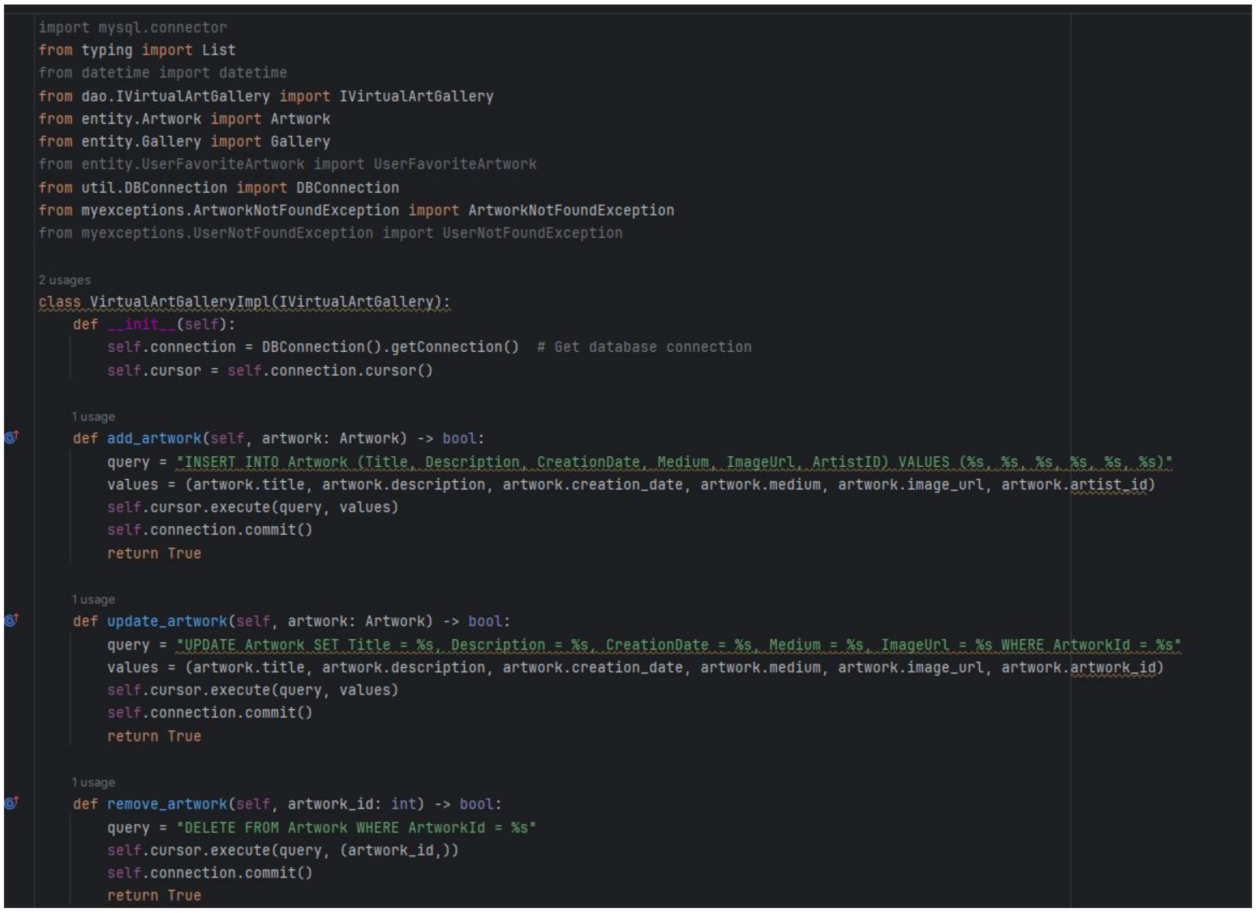


```python
from abc import ABC, abstractmethod
from typing import List
from entity.Artwork import Artwork
from entity.Gallery import Gallery

2 usages
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, artwork_id: int) -> bool:
        pass

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

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

    @abstractmethod
    def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
        pass

    @abstractmethod
    def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
        pass

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

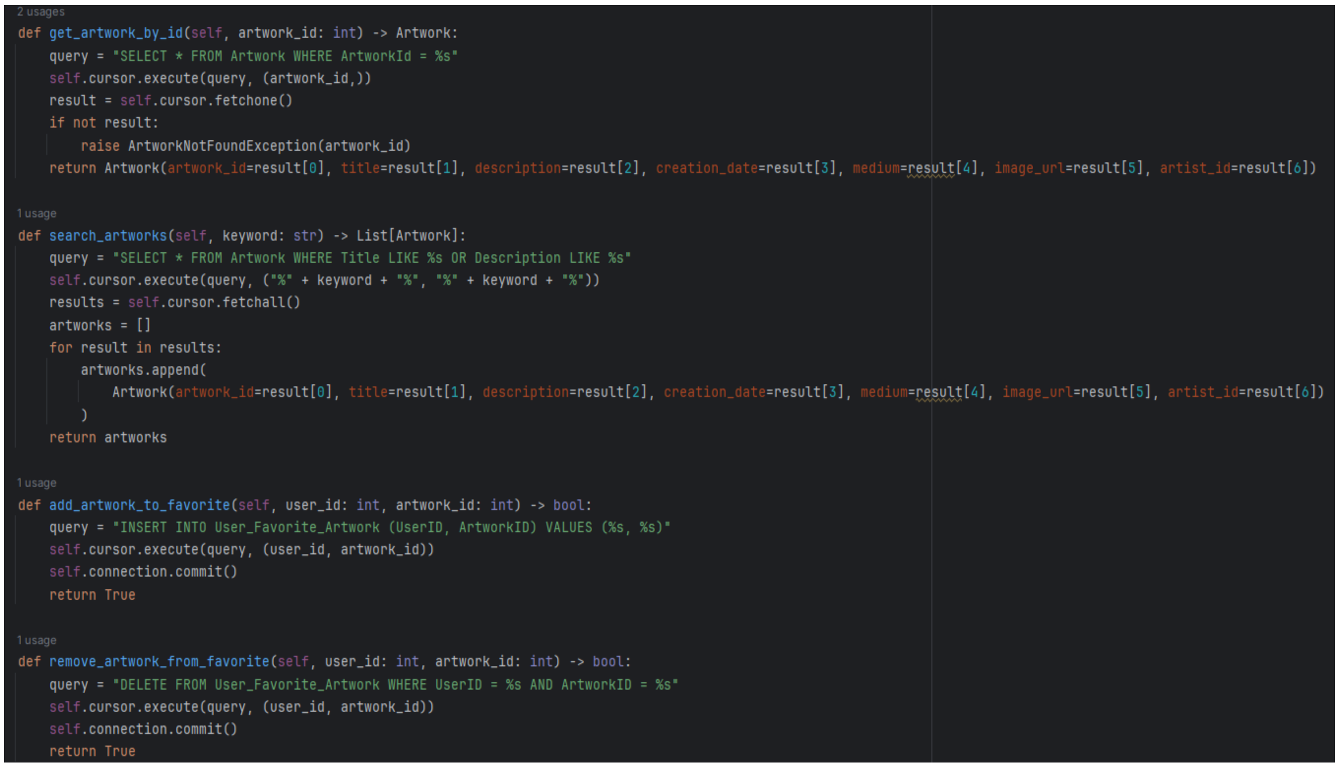


```python
    @abstractmethod
    def get_user_favorite_artworks(self, user_id: int) -> List[Artwork]:
        pass

    @abstractmethod
    def create_new_gallery(self, gallery: Gallery) -> bool:
        pass

    @abstractmethod
    def update_gallery(self, gallery: Gallery) -> bool:
        pass

    @abstractmethod
    def remove_gallery(self, gallery_id: int) -> bool:
        pass

    @abstractmethod
    def search_gallery(self, keyword: str) -> List[Gallery]:
        pass
```

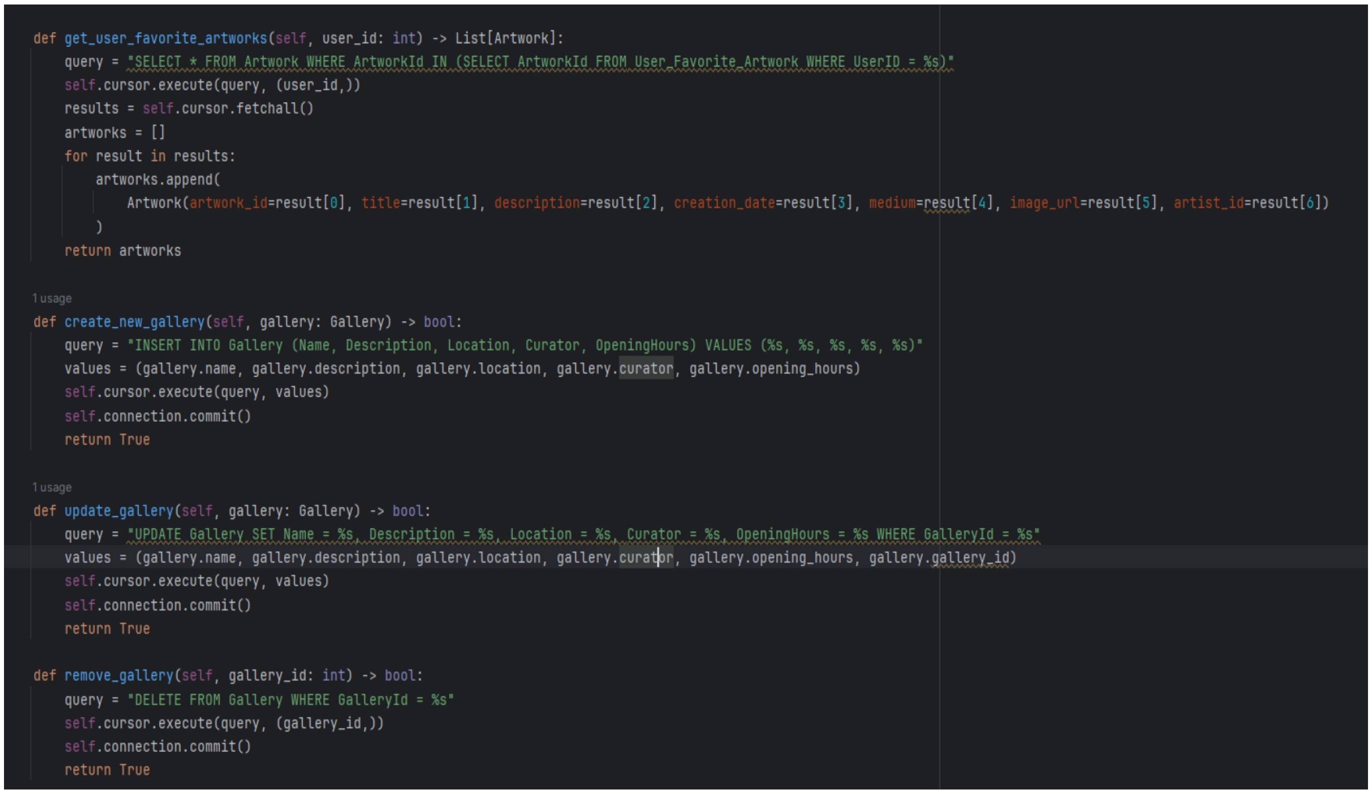
VirtualArtGalleryImpl.py



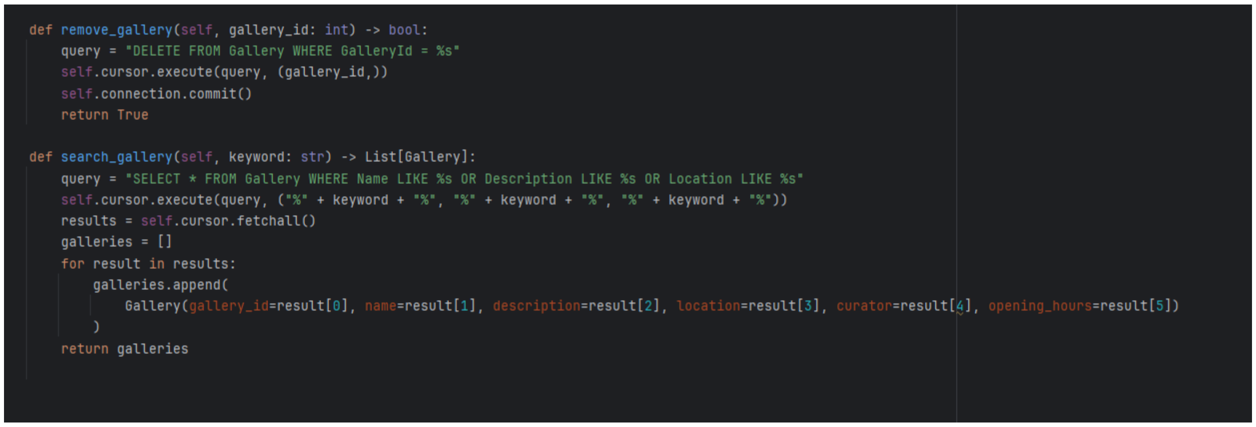
```python
import mysql.connector
from typing import List
from datetime import datetime
from dao.IVirtualArtGallery import IVirtualArtGallery
from entity.Artwork import Artwork
from entity.Gallery import Gallery
from entity.UserFavoriteArtwork import UserFavoriteArtwork
from util.DBConnection import DBConnection
from myexceptions.ArtworkNotFoundException import ArtworkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException

2 usages
class VirtualArtGalleryImpl(IVirtualArtGallery):
    def __init__(self):
        self.connection = DBConnection().getConnection()  # Get database connection
        self.cursor = self.connection.cursor()

    1 usage
    def add_artwork(self, artwork: Artwork) -> bool:
        query = "INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageUrl, ArtistID) VALUES (%s, %s, %s, %s, %s, %s)"
        values = (artwork.title, artwork.description, artwork.creation_date, artwork.medium, artwork.image_url, artwork.artist_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    1 usage
    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.title, artwork.description, artwork.creation_date, artwork.medium, artwork.image_url, artwork.artwork_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    1 usage
    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
```



```
2 usages
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 not result:
        raise ArtworkNotFoundException(artwork_id)
    return Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])

1 usage
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:
        artworks.append(
            Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])
        )
    return artworks

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

1 usage
def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
    query = "DELETE FROM User_Favorite_Artwork WHERE UserID = %s AND ArtworkID = %s"
    self.cursor.execute(query, (user_id, artwork_id))
    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:
        artworks.append(
            Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])
        )
    return artworks

1 usage
def create_new_gallery(self, gallery: Gallery) -> bool:
    query = "INSERT INTO Gallery (Name, Description, Location, Curator, OpeningHours) VALUES (%s, %s, %s, %s, %s)"
    values = (gallery.name, gallery.description, gallery.location, gallery.curator, gallery.opening_hours)
    self.cursor.execute(query, values)
    self.connection.commit()
    return True

1 usage
def update_gallery(self, gallery: Gallery) -> bool:
    query = "UPDATE Gallery SET Name = %s, Description = %s, Location = %s, Curator = %s, OpeningHours = %s WHERE GalleryId = %s"
    values = (gallery.name, gallery.description, gallery.location, gallery.curator, gallery.opening_hours, gallery.gallery_id)
    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
```



```python
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:
        galleries.append(
            Gallery(gallery_id=result[0], name=result[1], description=result[2], location=result[3], curator=result[4], opening_hours=result[5])
        )
    return galleries
```

7: Connect your application to the SQL database:

1. Write code to establish a connection to your SQL database.

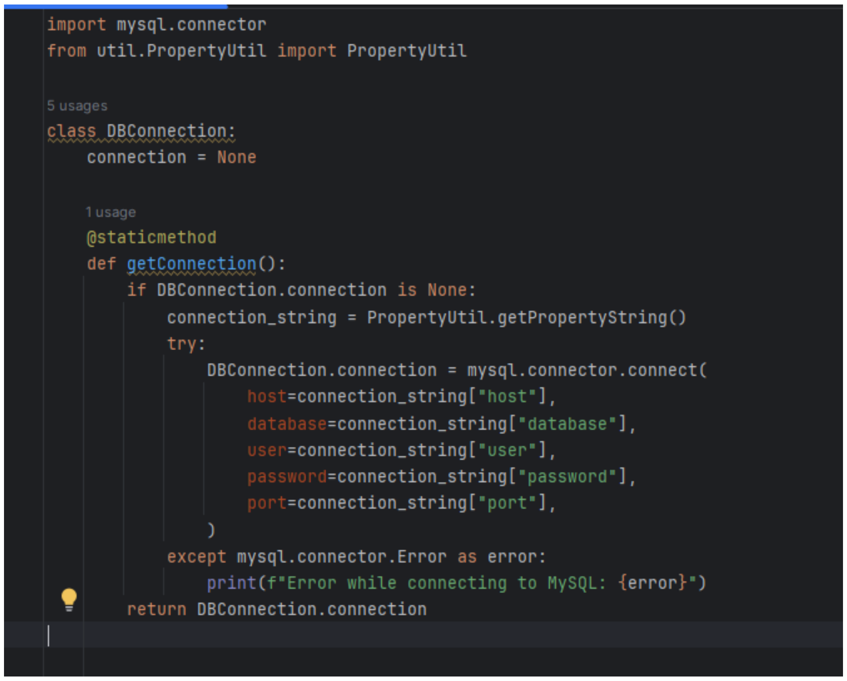
   Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

   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.

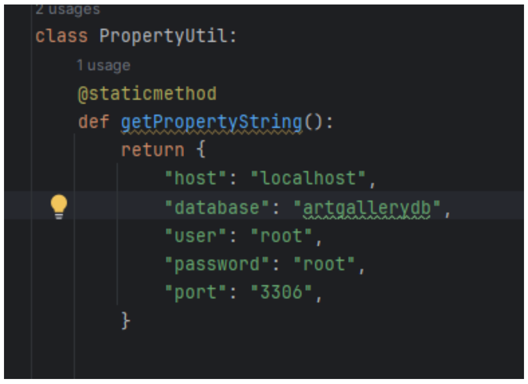
util package:

DBConnection.py:



```python
import mysql.connector
from util.PropertyUtil import PropertyUtil

5 usages
class DBConnection:
    connection = None

    1 usage
    @staticmethod
    def getConnection():
        if DBConnection.connection is None:
            connection_string = PropertyUtil.getPropertyString()
            try:
                DBConnection.connection = mysql.connector.connect(
                    host=connection_string["host"],
                    database=connection_string["database"],
                    user=connection_string["user"],
                    password=connection_string["password"],
                    port=connection_string["port"],
                )
            except mysql.connector.Error as error:
                print(f"Error while connecting to MySQL: {error}")
        return DBConnection.connection
```

PropertyUtil.py:

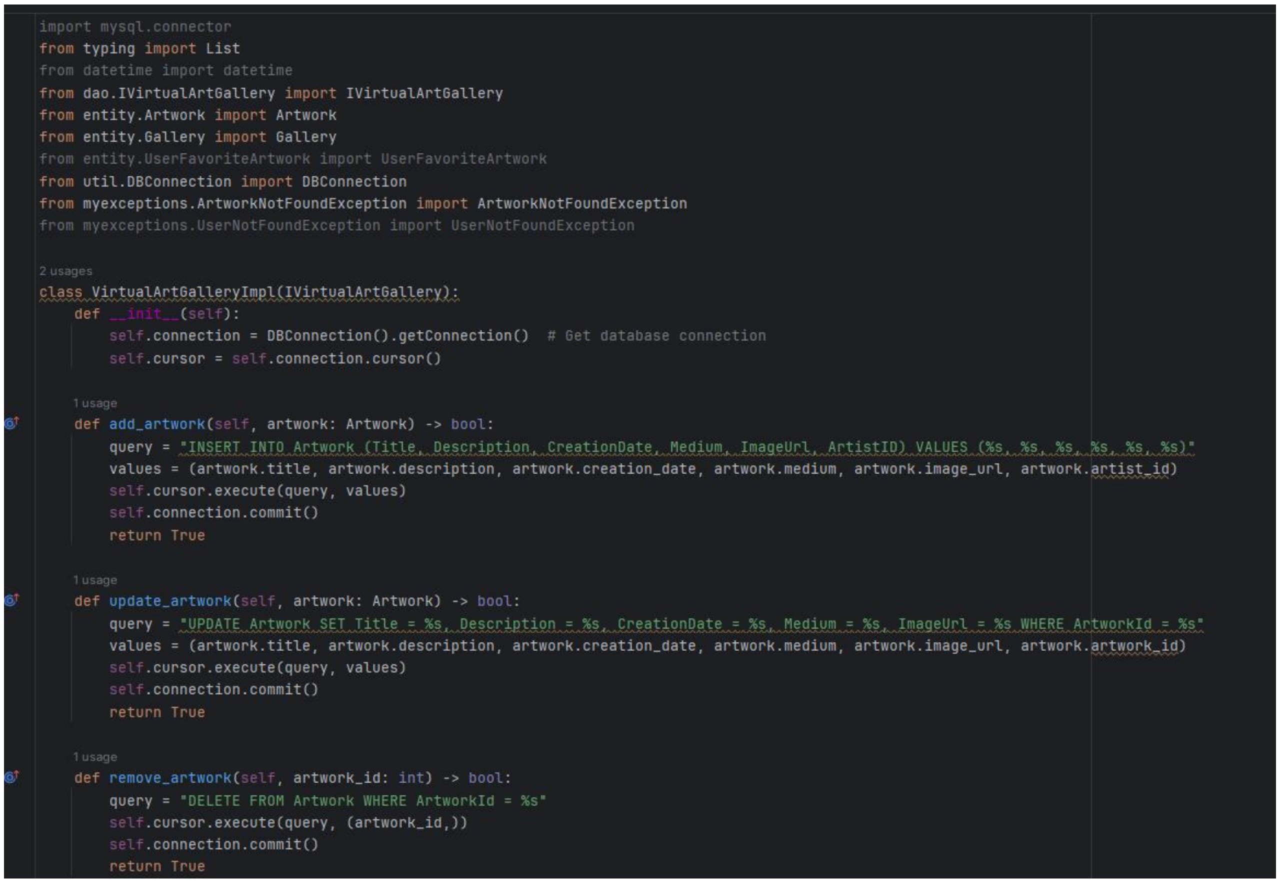


```
2 usages
class PropertyUtil:
    1 usage
    @staticmethod
    def getPropertyString():
        return {
            "host": "localhost",
            "database": "artgallerydb",
            "user": "root",
            "password": "root",
            "port": "3306",
        }
```

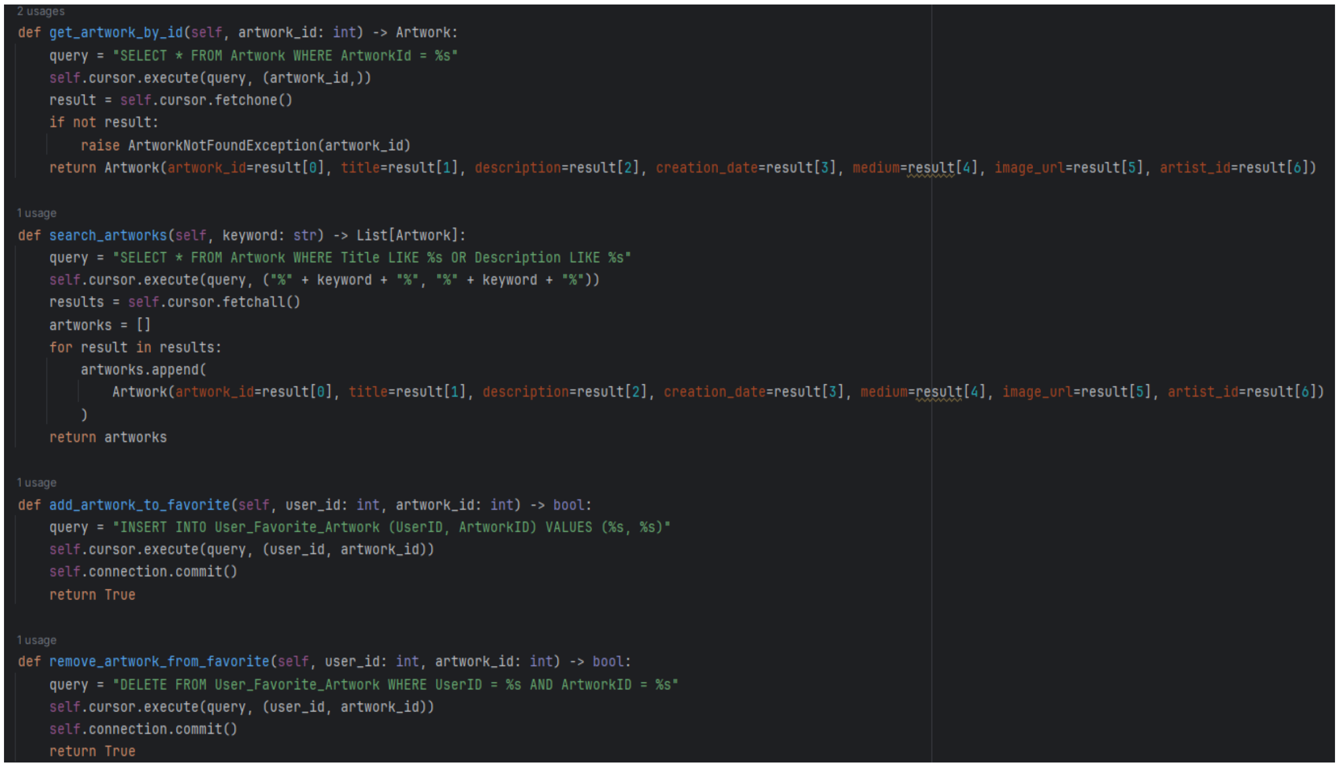
8: Service implementation

1. Create a Service class **VirtualArtGalleryImpl**
2. Provide implementation for all the methods in the interface.

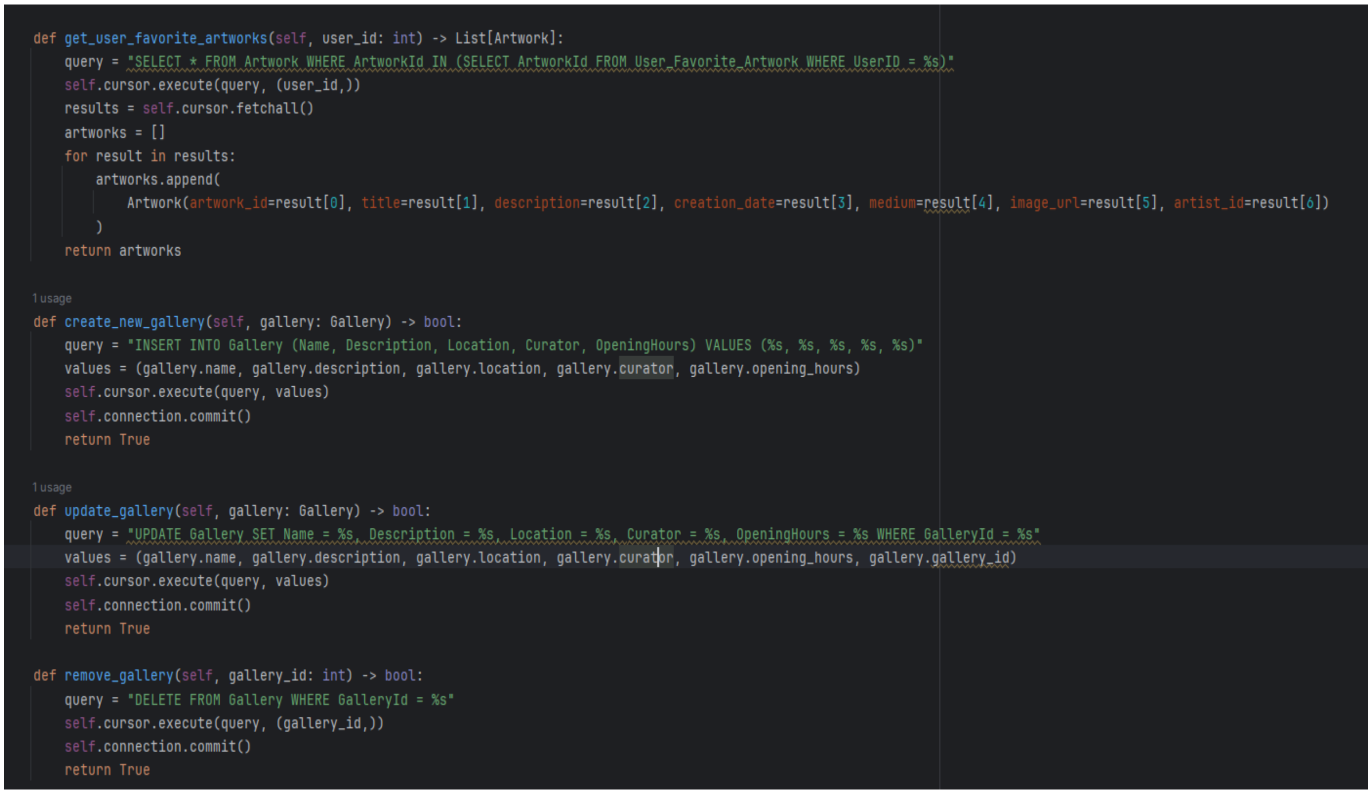
dao package: VirtualArtGalleryImpl.py:



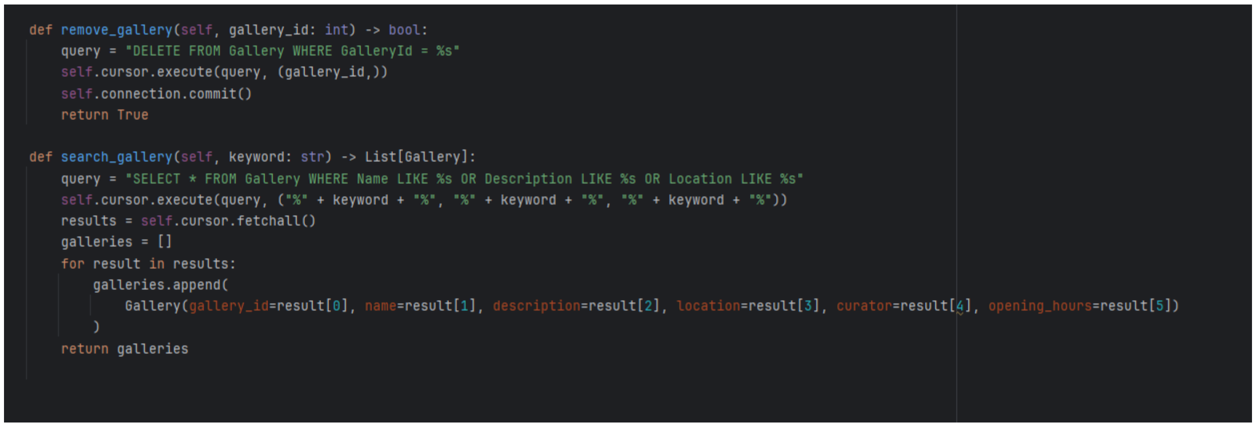
```
import mysql.connector
from typing import List
from datetime import datetime
from dao.IVirtualArtGallery import IVirtualArtGallery
from entity.Artwork import Artwork
from entity.Gallery import Gallery
from entity.UserFavoriteArtwork import UserFavoriteArtwork
from util.DBConnection import DBConnection
from myexceptions.ArtworkNotFoundException import ArtworkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException

2 usages
class VirtualArtGalleryImpl(IVirtualArtGallery):
    def __init__(self):
        self.connection = DBConnection().getConnection()  # Get database connection
        self.cursor = self.connection.cursor()

    1 usage
    def add_artwork(self, artwork: Artwork) -> bool:
        query = "INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageUrl, ArtistID) VALUES (%s, %s, %s, %s, %s, %s)"
        values = (artwork.title, artwork.description, artwork.creation_date, artwork.medium, artwork.image_url, artwork.artist_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    1 usage
    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.title, artwork.description, artwork.creation_date, artwork.medium, artwork.image_url, artwork.artwork_id)
        self.cursor.execute(query, values)
        self.connection.commit()
        return True

    1 usage
    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
```



```python
2 usages
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 not result:
        raise ArtworkNotFoundException(artwork_id)
    return Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])

1 usage
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:
        artworks.append(
            Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])
        )
    return artworks

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

1 usage
def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
    query = "DELETE FROM User_Favorite_Artwork WHERE UserID = %s AND ArtworkID = %s"
    self.cursor.execute(query, (user_id, artwork_id))
    self.connection.commit()
    return True
```



```python
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:
        artworks.append(
            Artwork(artwork_id=result[0], title=result[1], description=result[2], creation_date=result[3], medium=result[4], image_url=result[5], artist_id=result[6])
        )
    return artworks

1 usage
def create_new_gallery(self, gallery: Gallery) -> bool:
    query = "INSERT INTO Gallery (Name, Description, Location, Curator, OpeningHours) VALUES (%s, %s, %s, %s, %s)"
    values = (gallery.name, gallery.description, gallery.location, gallery.curator, gallery.opening_hours)
    self.cursor.execute(query, values)
    self.connection.commit()
    return True

1 usage
def update_gallery(self, gallery: Gallery) -> bool:
    query = "UPDATE Gallery SET Name = %s, Description = %s, Location = %s, Curator = %s, OpeningHours = %s WHERE GalleryId = %s"
    values = (gallery.name, gallery.description, gallery.location, gallery.curator, gallery.opening_hours, gallery.gallery_id)
    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
```



```python
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:
        galleries.append(
            Gallery(gallery_id=result[0], name=result[1], description=result[2], location=result[3], curator=result[4], opening_hours=result[5])
        )
    return galleries
```

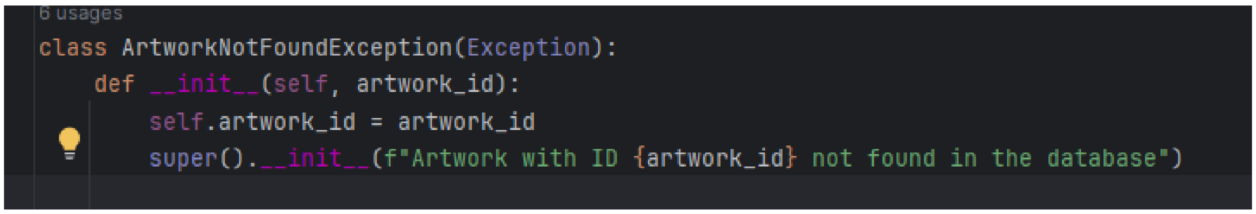
9: Exception Handling

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,

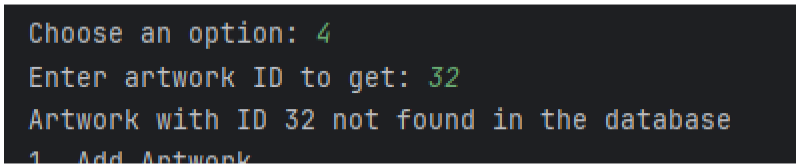
1. **ArtWorkNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db
2. **UserNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db

Myexceptions package:



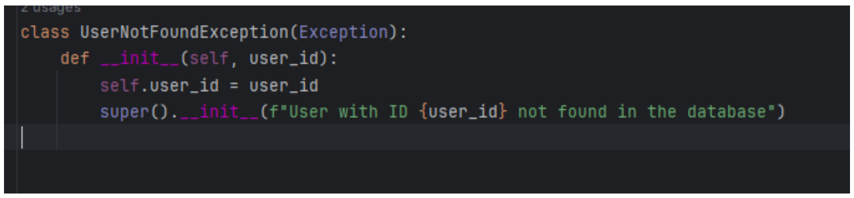
```python
6 usages
class ArtworkNotFoundException(Exception):
    def __init__(self, artwork_id):
        self.artwork_id = artwork_id
        super().__init__(f"Artwork with ID {artwork_id} not found in the database")
```

ArtWorkNotFoundException.py:

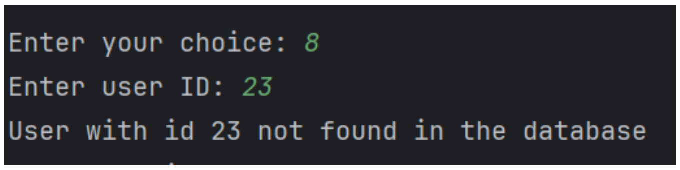


```
Choose an option: 4
Enter artwork ID to get: 32
Artwork with ID 32 not found in the database
```

UserNotFoundException.py:



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



```
Enter your choice: 8
Enter user ID: 23
User with id 23 not found in the database
```

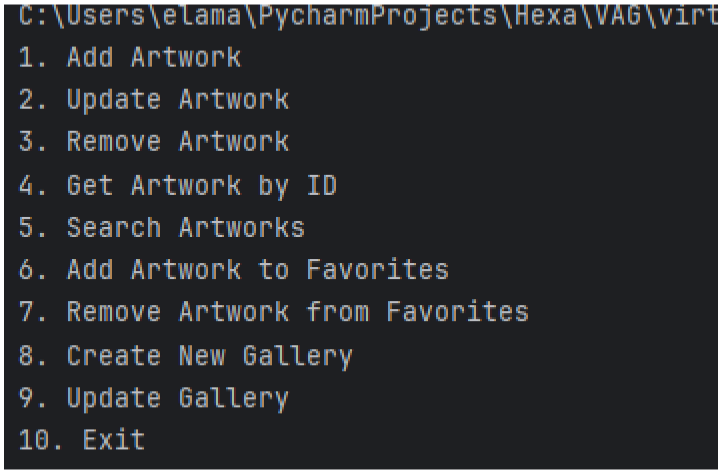
9. Main Method

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

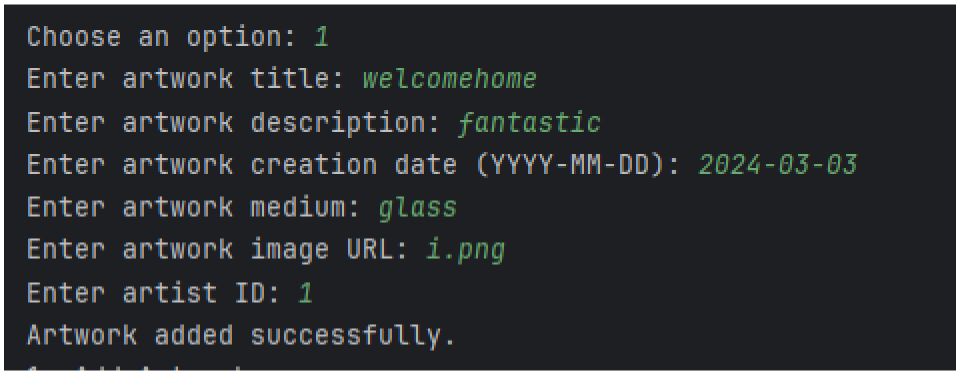
main package:

MainModule.py:

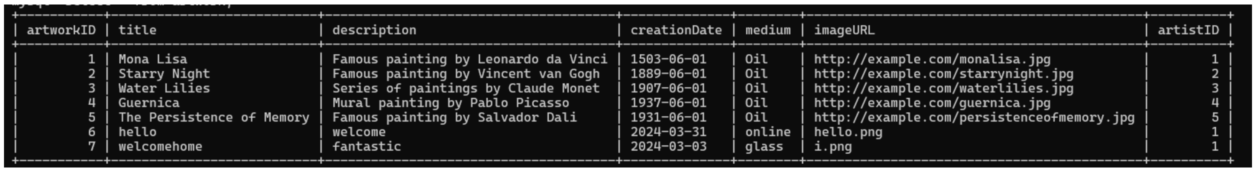
```python
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from entity.Artwork import Artwork
from entity.Gallery import Gallery
from myexceptions.ArtworkNotFoundException import ArtworkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException

class MainModule:
    def __init__(self):
        self.service = VirtualArtGalleryImpl()

    def menu(self):
        while True:
            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. Create New Gallery")
            print("9. Update Gallery")
            print("10. Exit")

            choice = input("Choose an option: ")

            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.create_new_gallery()
            elif choice == "9":
                self.update_gallery()
            elif choice == "10":
                print("Exiting Virtual Art Gallery")
                break
            else:
                print("Invalid option, please try again.")

    def add_artwork(self):
        title = input("Enter artwork title: ")
        description = input("Enter artwork description: ")
        creation_date = input("Enter artwork creation date (YYYY-MM-DD): ")
        medium = input("Enter artwork medium: ")
        image_url = input("Enter artwork image URL: ")
        artist_id = int(input("Enter artist ID: "))

        artwork = Artwork(
            title=title,
            description=description,
            creation_date=creation_date,
            medium=medium,
            image_url=image_url,
            artist_id=artist_id,
        )
        self.service.add_artwork(artwork)
        print("Artwork added successfully.")

    def update_artwork(self):
        artwork_id = int(input("Enter artwork ID to update: "))
        try:
            artwork = self.service.get_artwork_by_id(artwork_id)
            if artwork:
                title = input("Enter new title (press Enter to skip): ")
                description = input("Enter new description (press Enter to skip): ")
                creation_date = input("Enter new creation date (press Enter to skip): ")
                medium = input("Enter new medium (press Enter to skip): ")
                image_url = input("Enter new image URL (press Enter to skip): ")

                if title:
                    artwork.title = title
                if description:
                    artwork.description = description
                if creation_date:
                    artwork.creation_date = creation_date
                if medium:
                    artwork.medium = medium
                if image_url:
                    artwork.image_url = image_url

                self.service.update_artwork(artwork)
                print("Artwork updated successfully.")
            else:
                print("Artwork not found.")
        except ArtworkNotFoundException as e:
            print(e)

    def remove_artwork(self):
        artwork_id = int(input("Enter artwork ID to remove: "))
        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 to get: "))
        try:
            artwork = self.service.get_artwork_by_id(artwork_id)
            if artwork:
                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("Matching artworks:")
            for artwork in artworks:
                print(artwork)
        else:
            print("No matching 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_favorite(user_id, artwork_id)
        print("Artwork removed from favorites.")

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

        gallery = Gallery(
            gallery_id=gallery_id,
            name=name,
            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 gallery ID to update: "))
        try:
            gallery = self.service.get_gallery_by_id(gallery_id)
            if gallery:
                name = input("Enter new name (press Enter to skip): ")
                description = input("Enter new description (press Enter to skip): ")
                location = input("Enter new location (press Enter to skip): ")
                curator = input("Enter new curator (press Enter to skip): ")
                opening_hours = input("Enter new opening_hours (press Enter to skip): ")

                if name:
                    gallery.name = name
                if description:
                    gallery.description = description
                if location:
                    gallery.location = location
                if curator:
                    gallery.curator = curator
                if opening_hours:
                    gallery.opening_hours = opening_hours

                self.service.update_gallery(gallery)
                print("Gallery updated successfully.")
            else:
                print("Gallery not found.")
        except Exception as e:
            print(f"Error updating gallery: {e}")

if __name__ == "__main__":
    main_module = MainModule()
    main_module.menu()
```



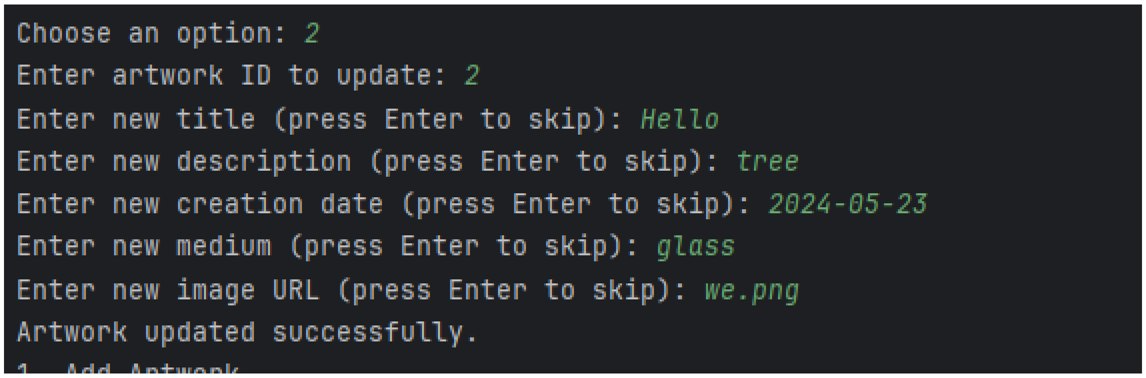
```
C:\Users\elama\PycharmProjects\Hexa\VAG\virt
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Create New Gallery
9. Update Gallery
10. Exit
```



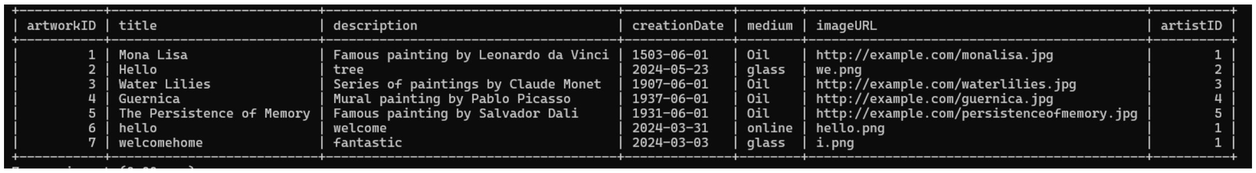
```
Choose an option: 1
Enter artwork title: welcomehome
Enter artwork description: fantastic
Enter artwork creation date (YYYY-MM-DD): 2024-03-03
Enter artwork medium: glass
Enter artwork image URL: i.png
Enter artist ID: 1
Artwork added successfully.
```



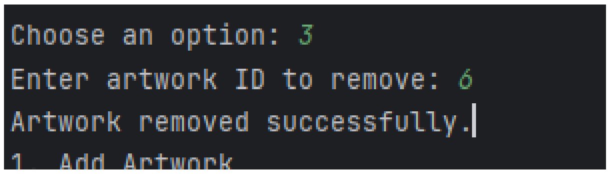
```
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
| artworkID | title                     | description                           | creationDate | medium | imageURL                                 | artistID |
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
|         1 | Mona Lisa                 | Famous painting by Leonardo da Vinci  | 1503-06-01   | Oil    | http://example.com/monalisa.jpg          |        1 |
|         2 | Starry Night              | Famous painting by Vincent van Gogh   | 1889-06-01   | Oil    | http://example.com/starrynight.jpg       |        2 |
|         3 | Water Lilies              | Series of paintings by Claude Monet   | 1907-06-01   | Oil    | http://example.com/waterlilies.jpg       |        3 |
|         4 | Guernica                  | Mural painting by Pablo Picasso       | 1937-06-01   | Oil    | http://example.com/guernica.jpg          |        4 |
|         5 | The Persistence of Memory | Famous painting by Salvador Dali      | 1931-06-01   | Oil    | http://example.com/persistenceofmemory.jpg |        5 |
|         6 | hello                     | welcome                               | 2024-03-31   | online | hello.png                                |        1 |
|         7 | welcomehome               | fantastic                             | 2024-03-03   | glass  | i.png                                    |        1 |
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
```



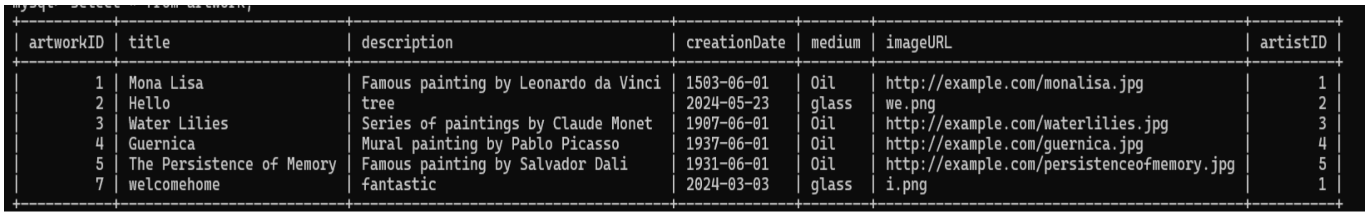
```
Choose an option: 2
Enter artwork ID to update: 2
Enter new title (press Enter to skip): Hello
Enter new description (press Enter to skip): tree
Enter new creation date (press Enter to skip): 2024-05-23
Enter new medium (press Enter to skip): glass
Enter new image URL (press Enter to skip): we.png
Artwork updated successfully.
1. Add Artwork
```



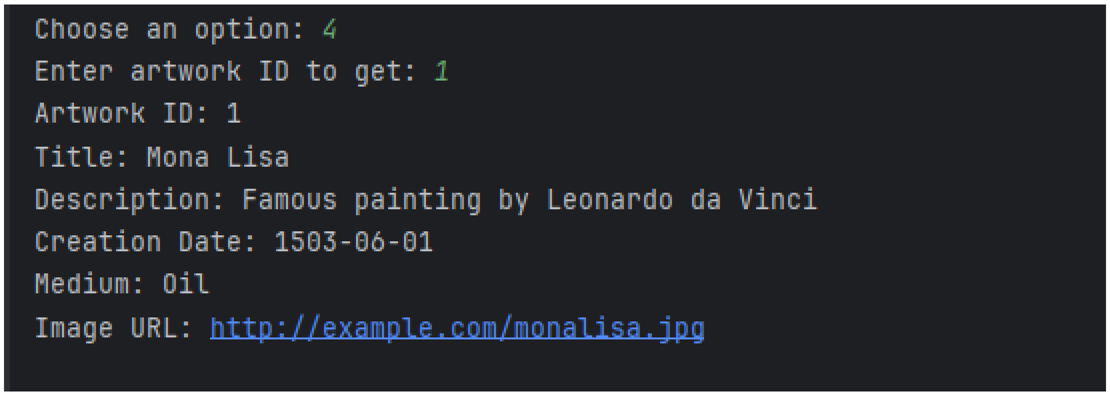
```
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
| artworkID | title                     | description                           | creationDate | medium | imageURL                                 | artistID |
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
|         1 | Mona Lisa                 | Famous painting by Leonardo da Vinci  | 1503-06-01   | Oil    | http://example.com/monalisa.jpg          |        1 |
|         2 | Hello                     | tree                                  | 2024-05-23   | glass  | we.png                                   |        2 |
|         3 | Water Lilies              | Series of paintings by Claude Monet   | 1907-06-01   | Oil    | http://example.com/waterlilies.jpg       |        3 |
|         4 | Guernica                  | Mural painting by Pablo Picasso       | 1937-06-01   | Oil    | http://example.com/guernica.jpg          |        4 |
|         5 | The Persistence of Memory | Famous painting by Salvador Dali      | 1931-06-01   | Oil    | http://example.com/persistenceofmemory.jpg |        5 |
|         6 | hello                     | welcome                               | 2024-03-31   | online | hello.png                                |        1 |
|         7 | welcomehome               | fantastic                             | 2024-03-03   | glass  | i.png                                    |        1 |
+-----------+---------------------------+---------------------------------------+--------------+--------+------------------------------------------+----------+
```



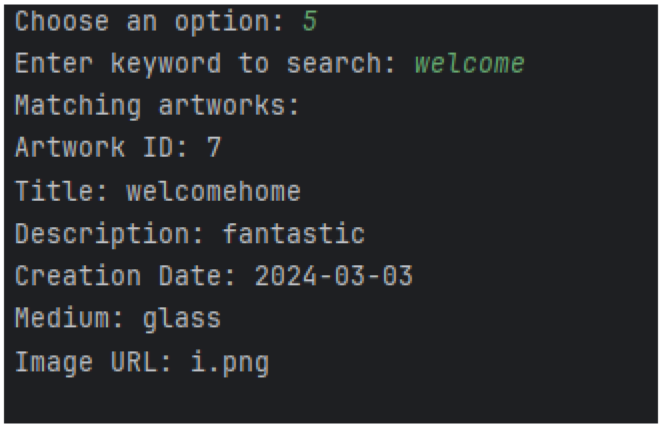
```
Choose an option: 3
Enter artwork ID to remove: 6
Artwork removed successfully.
1. Add Artwork
```



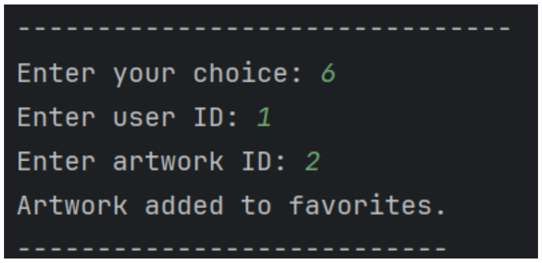
```
+-----------+---------------------------+-------------------------------------+--------------+--------+-----------------------------------------------+----------+
| artworkID | title                     | description                         | creationDate | medium | imageURL                                      | artistID |
+-----------+---------------------------+-------------------------------------+--------------+--------+-----------------------------------------------+----------+
|         1 | Mona Lisa                 | Famous painting by Leonardo da Vinci | 1503-06-01   | Oil    | http://example.com/monalisa.jpg               |        1 |
|         2 | Hello                     | tree                                | 2024-05-23   | glass  | we.png                                        |        2 |
|         3 | Water Lilies              | Series of paintings by Claude Monet | 1907-06-01   | Oil    | http://example.com/waterlilies.jpg            |        3 |
|         4 | Guernica                  | Mural painting by Pablo Picasso     | 1937-06-01   | Oil    | http://example.com/guernica.jpg               |        4 |
|         5 | The Persistence of Memory | Famous painting by Salvador Dali    | 1931-06-01   | Oil    | http://example.com/persistenceofmemory.jpg    |        5 |
|         7 | welcomehome               | fantastic                           | 2024-03-03   | glass  | i.png                                         |        1 |
+-----------+---------------------------+-------------------------------------+--------------+--------+-----------------------------------------------+----------+
```



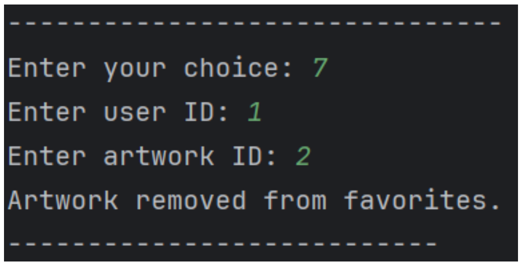
```
Choose an option: 4
Enter artwork ID to get: 1
Artwork ID: 1
Title: Mona Lisa
Description: Famous painting by Leonardo da Vinci
Creation Date: 1503-06-01
Medium: Oil
Image URL: http://example.com/monalisa.jpg
```



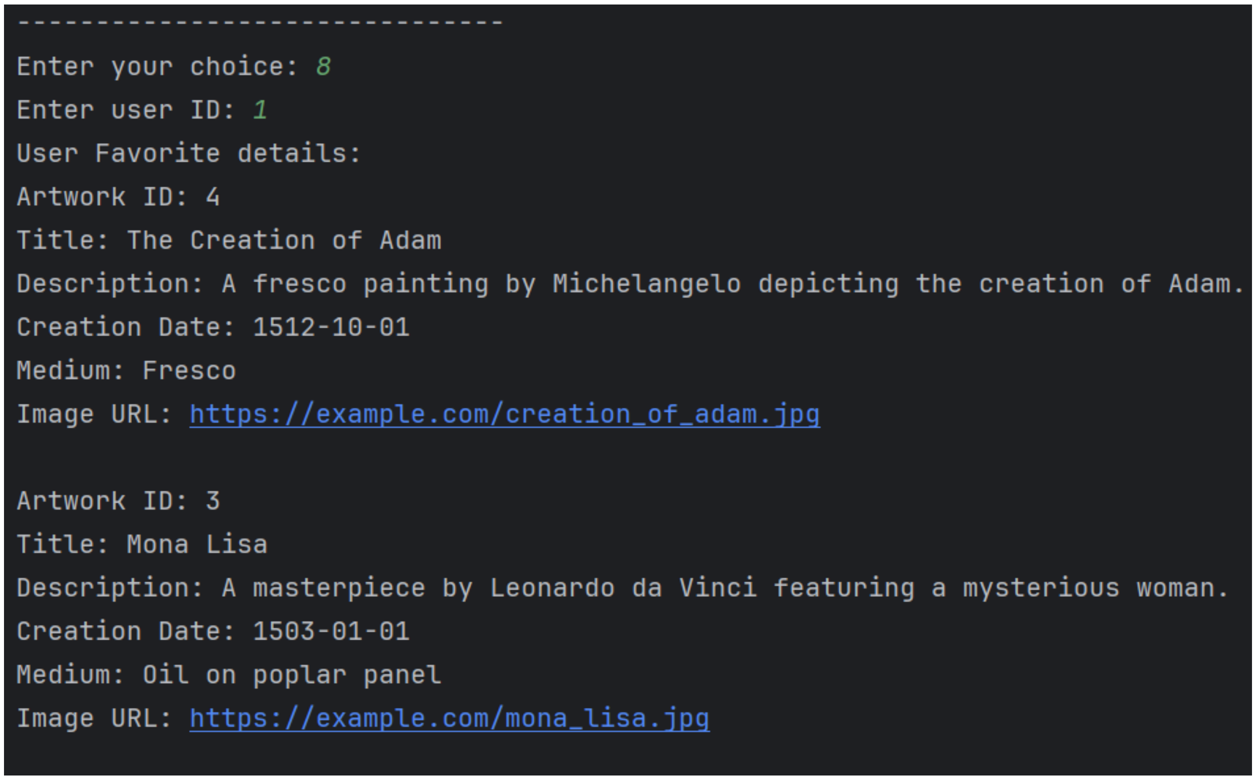
```
Choose an option: 5
Enter keyword to search: welcome
Matching artworks:
Artwork ID: 7
Title: welcomehome
Description: fantastic
Creation Date: 2024-03-03
Medium: glass
Image URL: i.png
```



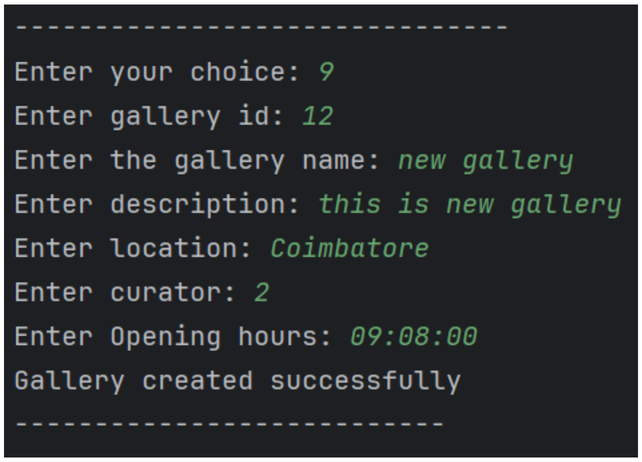
```
-------------------------------
Enter your choice: 6
Enter user ID: 1
Enter artwork ID: 2
Artwork added to favorites.
---------------------------
```



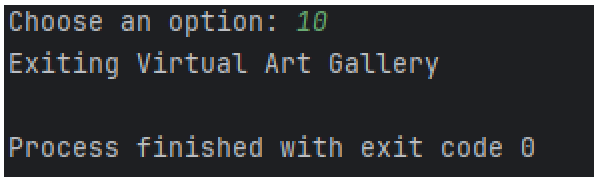
```
------------------------------
Enter your choice: 7
Enter user ID: 1
Enter artwork ID: 2
Artwork removed from favorites.
--------------------------
```



```
------------------------------
Enter your choice: 8
Enter user ID: 1
User Favorite details:
Artwork ID: 4
Title: The Creation of Adam
Description: A fresco painting by Michelangelo depicting the creation of Adam.
Creation Date: 1512-10-01
Medium: Fresco
Image URL: https://example.com/creation_of_adam.jpg

Artwork ID: 3
Title: Mona Lisa
Description: A masterpiece by Leonardo da Vinci featuring a mysterious woman.
Creation Date: 1503-01-01
Medium: Oil on poplar panel
Image URL: https://example.com/mona_lisa.jpg
```



```
------------------------------
Enter your choice: 9
Enter gallery id: 12
Enter the gallery name: new gallery
Enter description: this is new gallery
Enter location: Coimbatore
Enter curator: 2
Enter Opening hours: 09:08:00
Gallery created successfully
--------------------------
```



```
Choose an option: 10
Exiting Virtual Art Gallery

Process finished with exit code 0
```

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.

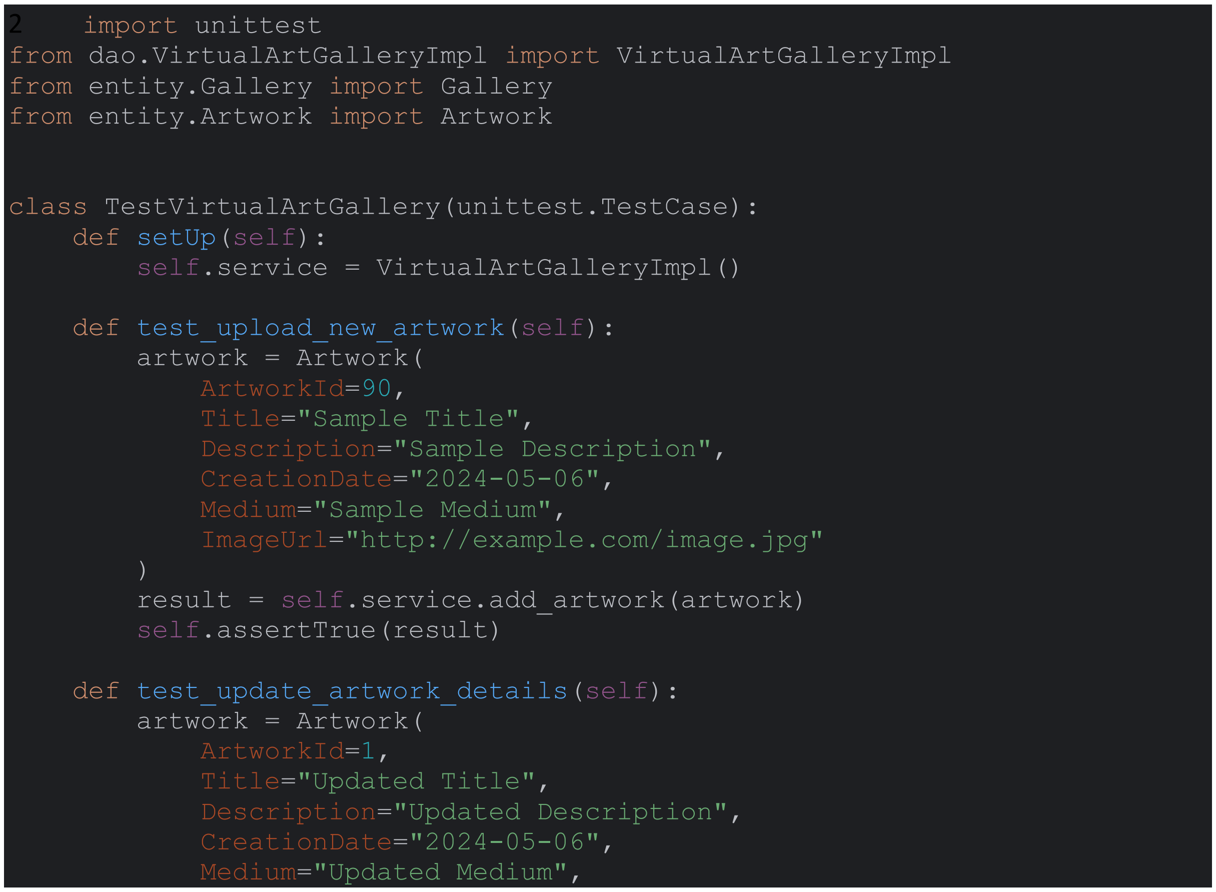
2. **Gallery Management:**

a. Test creating a new gallery.

b. Verify that updating gallery information works correctly.

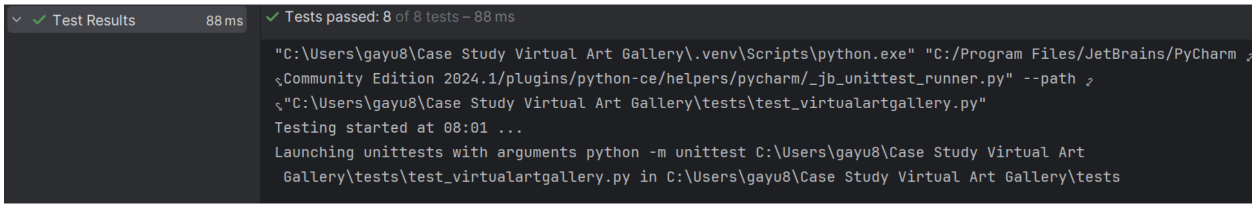
c. Test removing a gallery from the system.

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



```
import unittest
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from entity.Gallery import Gallery
from entity.Artwork import Artwork


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

    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",
```

```python
            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 = "Sample"
        artworks = self.service.search_artworks(keyword)
        self.assertTrue(len(artworks) > 0)

    def test_create_new_gallery(self):
        gallery = Gallery(
            GalleryId=1,
            Name="Sample Gallery",
            Description="Sample Description",
            Location="Sample Location",
            Curator=1,
            OpeningHours="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,
            OpeningHours="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 = "House"
        galleries = self.service.search_gallery(keyword)
        self.assertTrue(len(galleries) > 0)


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



```
"C:\Users\gayu8\Case Study Virtual Art Gallery\.venv\Scripts\python.exe" "C:/Program Files/JetBrains/PyCharm
Community Edition 2024.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --path
"C:\Users\gayu8\Case Study Virtual Art Gallery\tests\test_virtualartgallery.py"
Testing started at 08:01 ...
Launching unittests with arguments python -m unittest C:\Users\gayu8\Case Study Virtual Art
 Gallery\tests\test_virtualartgallery.py in C:\Users\gayu8\Case Study Virtual Art Gallery\tests
```