Coding Challenges: PetPals, The Pet Adoption Platform

Implement OOPs

Create SQL Schema from the pet and user class, use the class attributes for table column names. 1.Create and implement the mentioned class and the structure in your application.

Pet Class:

Attributes: • Name (string): The name of the pet. • Age (int): The age of the pet. • Breed (string): The breed of the pet.

Methods: • Constructor to initialize Name, Age, and Breed. • Getters and setters for attributes. • ToString() method to provide a string representation of the pet.

```python
from abc import ABC,abstractmethod
class Pet:
    def __init__(self, name, age, breed):
        self.name = name
        self.age = age
        self.breed = breed

    def __str__(self):
        return f"Name: {self.name}, Age: {self.age}, Breed: {self.breed}"
    @property
    def getName(self):
        return self.name

    @getName.setter
    def setName(self,newName):
        self.name = newName

    @property
    def getAge(self):
        return self.age

    @getAge.setter
    def setName(self, newAge):
        self.age = newAge

@property
def getBreed(self):
    return self.breed

@getBreed.setter
def setBreed(self, newBreed):
    self.breed = newBreed
```

Dog Class (Inherits from Pet):

Additional Attributes: • DogBreed (string): The specific breed of the dog. Additional Methods: • Constructor to initialize DogBreed. • Getters and setters for DogBreed.

```
35  class Dog(Pet):
36      def __init__(self, name, age, breed, dogBreed):
37          super().__init__(name, age, breed)
38          self.dogBreed = dogBreed
39
40      @property
41      def getDogBreed(self):
42          return self.dogBreed
43
44      @getDogBreed.setter
45      def setDogBreed(self,newDogBreed):
46          self.dogBreed = newDogBreed
47
```

Cat Class (Inherits from Pet):

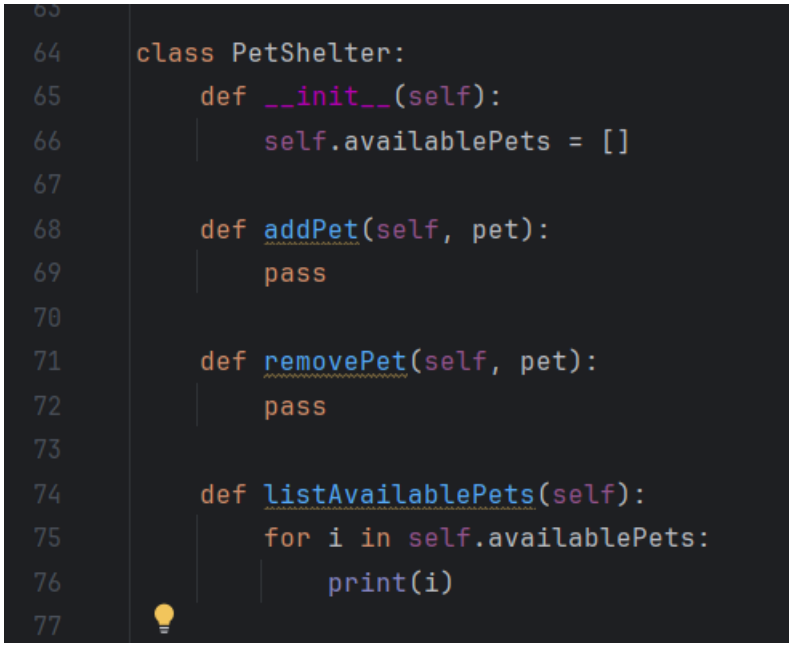
Additional Attributes: • CatColor (string): The color of the cat. Additional Methods: • Constructor to initialize CatColor. • Getters and setters for CatColor.

```
50  class Cat(Pet):
51      def __init__(self, name, age, breed, catColor):
52          super().__init__(name, age, breed)
53          self.catColor = catColor
54
55      @property
56      def getCatColor(self):
57          return self.catColor
58
59      @getCatColor.setter
60      def setCatColor(self,newCatColor):
61          self.catColor = newCatColor
62
```

3.PetShelter Class:

Attributes: • availablePets (List of Pet): A list to store available pets for adoption.

Methods: • AddPet(Pet pet): Adds a pet to the list of available pets. • RemovePet(Pet pet): Removes a pet from the list of available pets. • ListAvailablePets(): Lists all available pets in the shelter.

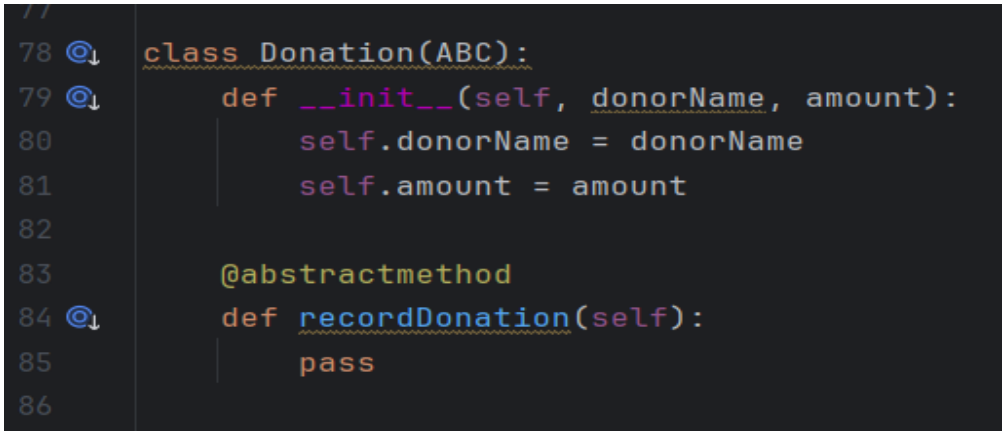


```
class PetShelter:
    def __init__(self):
        self.availablePets = []

    def addPet(self, pet):
        pass

    def removePet(self, pet):
        pass

    def listAvailablePets(self):
        for i in self.availablePets:
            print(i)
```

4.Donation Class (Abstract):

Attributes:• DonorName (string): The name of the donor. • Amount (decimal): The donation amount.

Methods: • Constructor to initialize DonorName and Amount. • Abstract method RecordDonation() to record the donation (to be implemented in derived classes).



```
class Donation(ABC):
    def __init__(self, donorName, amount):
        self.donorName = donorName
        self.amount = amount

    @abstractmethod
    def recordDonation(self):
        pass
```

CashDonation Class (Derived from Donation):

Additional Attributes: • DonationDate (DateTime): The date of the cash donation. Additional Methods: • Constructor to initialize DonationDate. • Implementation of RecordDonation() to record a cash donation.

```
class CashDonation(Donation):
    def __init__(self, donorName, amount, donationDate):
        super().__init__(donorName, amount)
        self.donationDate = donationDate

    def recordDonation(self):
        print(f"Name of the donor is {self.donorName} , Amount donated {self.amount} on {self.donationDate}")
```

ItemDonation Class (Derived from Donation):

Additional Attributes: • ItemType (string): The type of item donated (e.g., food, toys). Additional Methods: • Constructor to initialize ItemType. • Implementation of RecordDonation() to record an item donation.

```
class ItemDonation(Donation):
    def __init__(self, donorName, amount, itemType):
        super().__init__(donorName, amount)
        self.itemType = itemType

    def recorDonation(self):
        print(f"Item donation of {self.itemType} recorded.")
```

5.IAdoptable Interface/Abstract Class:

Methods: • Adopt(): An abstract method to handle the adoption process.

AdoptionEvent Class:

Attributes: • Participants (List of IAdoptable): A list of participants (shelters and adopters) in the adoption event.

Methods: • HostEvent(): Hosts the adoption event. • RegisterParticipant(IAdoptable participant): Registers a participant for the event.

```
105
106     class IAdoptable(ABC):
107
108         @abstractmethod
109         def adopt(self):
110             pass
111
112     class AdoptionEvent:
113         def __init__(self):
114             self.participants = []
115
116         def hostEvent(self):
117             print("Adoption Hosting.")
118
119         def registerParticipant(self, participant):
120             self.participants = participant
121
```

7.Database Connectivity

Create and implement the following tasks in your application.

- **Displaying Pet Listings:**

o Develop a program that connects to the database and retrieves a list of available pets from the "pets" table. Display this list to the user. Ensure that the program handles database connectivity exceptions gracefully, including cases where the database is unreachable.

```python
import mysql.connector

class databaseconnction():
    def __init__(self,host,user,password,database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connection = None

    def connect(self):
        try:
            self.connection = mysql.connector.connect(
            host = self.host,
            user = self.user,
            password = self.password,
            database = self.database
            )
            print(f"Connected to {self.database} database")
        except:
            print("Could not connect to database")

    def disconnect(self):
        if self.connection:
            self.connection.close()
            print("Disconnected")

```

```python
    def AvailablePets(self):
        cursor = self.connection.cursor()
        cursor.execute("select * from pets where AvailableForAdoption = %s",(1,))
        result = cursor.fetchall()
        cursor.close()
        for i in result:
            print(i)
```

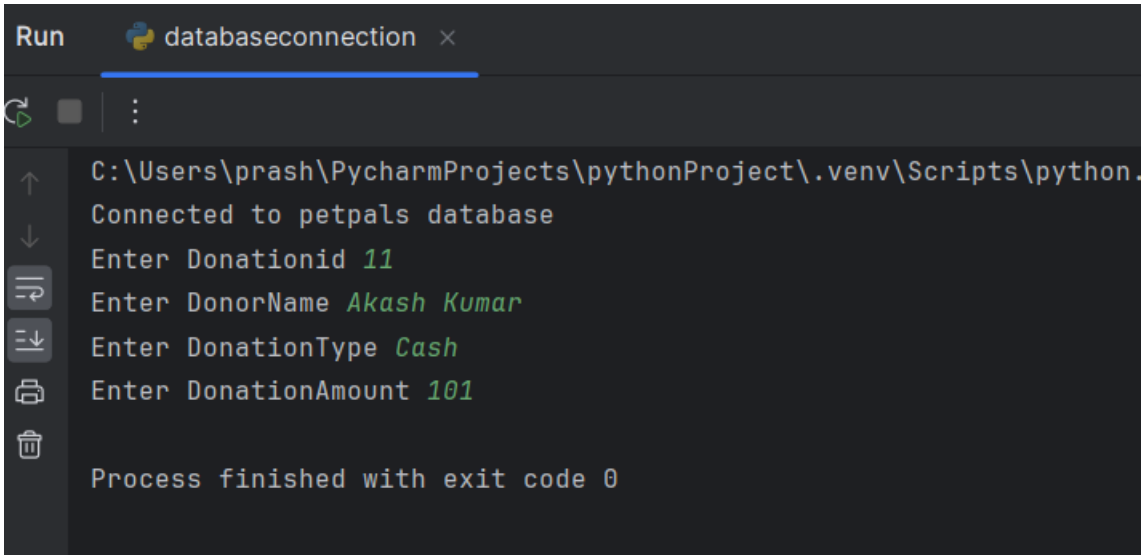
```python
if __name__ == "__main__":

    a = databaseconnction( host: "localhost", user: "root", password: [redacted] database: "petpals")
    a.connect()
    a.AvailablePets()
```

```
if __name__ == "__main__"
Run  databaseconnection
C:\Users\prash\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\prash\PycharmProjects\py
Connected to petpals database
(1, 'Tommy', 4, 'golden retriever', 'Dog', 1, None)
(2, 'Buddy', 3, 'Labrador', 'Dog', 1, None)
(3, 'Whiskers', 2, 'Siamese', 'Cat', 1, None)
(5, 'Fluffy', 1, 'Persian', 'Cat', 1, None)
(6, 'Milo', 2, 'Beagle', 'Dog', 1, None)
(7, 'Luna', 1, 'Ragdoll', 'Cat', 1, None)
(9, 'Whiskey', 3, 'Scottish Fold', 'Cat', 1, None)
(10, 'Charlie', 4, 'Dachshund', 'Dog', 1, None)

Process finished with exit code 0
```

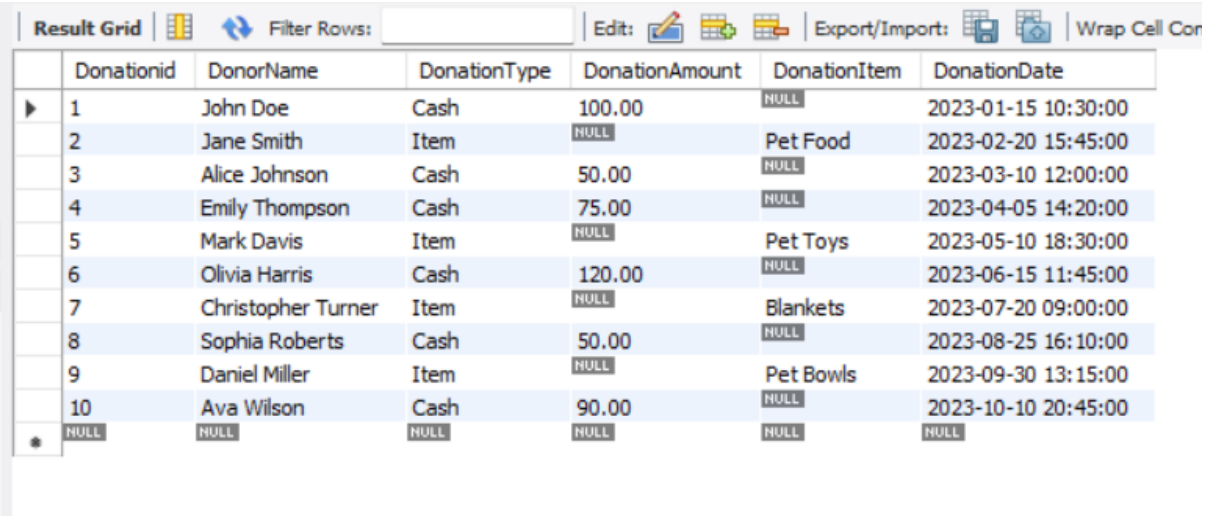
- **Donation Recording:**

o Create a program that records cash donations made by donors. Allow the user to input donor information and the donation amount and insert this data into the "donations" table in the database. Handle exceptions related to database operations, such as database errors or invalid inputs.

```python
    def donate(self):
        cursor = self.connection.cursor()

        Donationid = int(input("Enter Donationid "))
        DonorName = input("Enter DonorName ")
        DonationType = input("Enter DonationType ")
        DonationAmount = int(input("Enter DonationAmount "))

        cursor.execute("insert into donations(Donationid,DonorName,DonationType,DonationAmount) "
                       "values(%s,%s,%s,%s)",( Donationid,DonorName,DonationType,DonationAmount) )
        self.connection.commit()
        cursor.close()
```



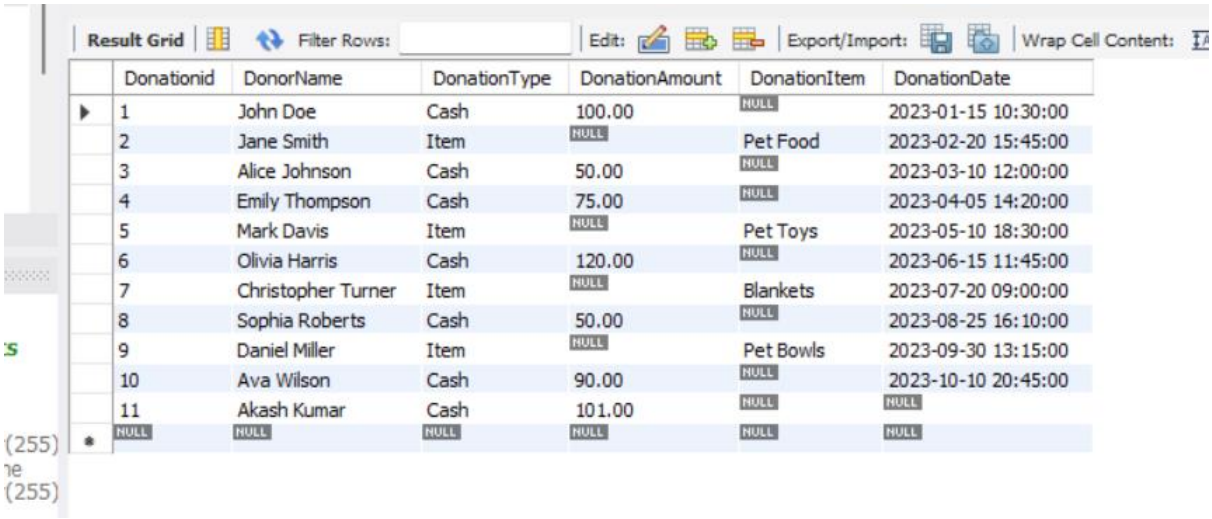
After



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

Donationid	DonorName	DonationType	DonationAmount	DonationItem	DonationDate
1	John Doe	Cash	100.00	NULL	2023-01-15 10:30:00
2	Jane Smith	Item	NULL	Pet Food	2023-02-20 15:45:00
3	Alice Johnson	Cash	50.00	NULL	2023-03-10 12:00:00
4	Emily Thompson	Cash	75.00	NULL	2023-04-05 14:20:00
5	Mark Davis	Item	NULL	Pet Toys	2023-05-10 18:30:00
6	Olivia Harris	Cash	120.00	NULL	2023-06-15 11:45:00
7	Christopher Turner	Item	NULL	Blankets	2023-07-20 09:00:00
8	Sophia Roberts	Cash	50.00	NULL	2023-08-25 16:10:00
9	Daniel Miller	Item	NULL	Pet Bowls	2023-09-30 13:15:00
10	Ava Wilson	Cash	90.00	NULL	2023-10-10 20:45:00
NULL	NULL	NULL	NULL	NULL	NULL

Before



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

Donationid	DonorName	DonationType	DonationAmount	DonationItem	DonationDate
1	John Doe	Cash	100.00	NULL	2023-01-15 10:30:00
2	Jane Smith	Item	NULL	Pet Food	2023-02-20 15:45:00
3	Alice Johnson	Cash	50.00	NULL	2023-03-10 12:00:00
4	Emily Thompson	Cash	75.00	NULL	2023-04-05 14:20:00
5	Mark Davis	Item	NULL	Pet Toys	2023-05-10 18:30:00
6	Olivia Harris	Cash	120.00	NULL	2023-06-15 11:45:00
7	Christopher Turner	Item	NULL	Blankets	2023-07-20 09:00:00
8	Sophia Roberts	Cash	50.00	NULL	2023-08-25 16:10:00
9	Daniel Miller	Item	NULL	Pet Bowls	2023-09-30 13:15:00
10	Ava Wilson	Cash	90.00	NULL	2023-10-10 20:45:00
11	Akash Kumar	Cash	101.00	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

- **Adoption Event Management:**

o Build a program that connects to the database and retrieves information about upcoming adoption events from the "adoption_events" table. Allow the user to register for an event by adding their details to the "participants" table. Ensure that the program handles database connectivity and insertion exceptions properly.

```
49      def events(self):
50          cursor = self.connection.cursor()
51          cursor.execute("select * from adoptionevents")
52          result = cursor.fetchall()
53          cursor.close()
54          for i in result:
55              print(i)
```

```
74  if __name__ == "__main__":
75
76      a = databaseconnction( host: "localhost", user: "root", password: [redacted], database: "petpals")
77      a.connect()
78      a.events()
```

```
if __name__ == "__main__"
Run    databaseconnection
C:\Users\prash\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\prash\PycharmProjects\pyt
Connected to petpals database
(1, 'PetPal Adoption Day', datetime.datetime(2023, 4, 1, 13, 0), 'Central Park')
(2, 'Rescue Fair', datetime.datetime(2023, 5, 15, 11, 0), 'Town Hall Plaza')
(3, 'Furry Fiesta Adoption Day', datetime.datetime(2023, 11, 5, 12, 0), 'City Park')
(4, 'Home for the Holidays', datetime.datetime(2023, 12, 15, 14, 30), 'Winter Wonderland Hall')
(5, 'Spring Fling Pet Adoption Fair', datetime.datetime(2024, 3, 20, 11, 0), 'Sunny Meadows Park')
(6, 'Summer Splash Adoption Event', datetime.datetime(2024, 7, 1, 13, 30), 'Beachside Pavilion')
(7, 'Autumn Paws Festival', datetime.datetime(2024, 9, 15, 10, 45), 'Harvest Grove Community Center')
(8, 'Spooky Pet Parade and Adoption', datetime.datetime(2024, 10, 31, 17, 0), 'Haunted Haven Square')
(9, 'Winter Whiskers Wonderland', datetime.datetime(2024, 12, 20, 15, 0), 'Crystal Ice Palace')
(10, 'New Furry Friend', datetime.datetime(2025, 1, 5, 12, 15), 'Celebration Plaza')

Process finished with exit code 0
```

```python

    def participate(self):
        cursor = self.connection.cursor()
        ParticipantID = int(input("Enter ParticipantID "))
        ParticipantName = input("Enter  ParticipantName ")
        ParticipantType = input("Enter ParticipantType ")
        EventID = int(input("Enter EventID "))

        cursor.execute("insert into participants(ParticipantID,ParticipantName,ParticipantType,EventID) "
                       "values (%s,%s,%s,%s)",(ParticipantID,ParticipantName,ParticipantType,EventID))
        self.connection.commit()
        cursor.close()

```

```python
if __name__ == "__main__":

    a = databaseconnction( host: "localhost", user: "root", password: [redacted] database: "petpals")
    a.connect()
    # a.donate()
    # a.events()
    a.participate()
```

Run databaseconnection

```
C:\Users\prash\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\prash\PycharmProjects\pyth
Connected to petpals database
Enter ParticipantID 11
Enter  ParticipantName "Akash Kumar"
Enter ParticipantType "Adopter"
Enter EventID 2
```

Before

ParticipantID	ParticipantName	ParticipantType	EventID	adoptedPetId
1	Paws Haven	Shelter	1	1
2	Adopter123	Adopter	1	2
3	Rescue Me Shelter	Shelter	2	3
4	Paws of Love	Shelter	3	5
5	Adopter456	Adopter	3	6
6	Rescue Rangers	Shelter	4	7
7	PetLovers101	Adopter	4	9
8	Cuddly Companions	Shelter	5	10
9	FurryFamily	Adopter	5	NULL
10	Purrfect Partners	Shelter	6	NULL
NULL	NULL	NULL	NULL	NULL

After

Result Grid | Filter Rows: | Edit: | Export/Impo

ParticipantID	ParticipantName	ParticipantType	EventID	adoptedPetId
1	Paws Haven	Shelter	1	1
2	Adopter123	Adopter	1	2
3	Rescue Me Shelter	Shelter	2	3
4	Paws of Love	Shelter	3	5
5	Adopter456	Adopter	3	6
6	Rescue Rangers	Shelter	4	7
7	PetLovers101	Adopter	4	9
8	Cuddly Companions	Shelter	5	10
9	FurryFamily	Adopter	5	NULL
10	Purrfect Partners	Shelter	6	NULL
11	"Akash Kumar"	"Adopter"	2	NULL
NULL	NULL	NULL	NULL	NULL

6.Exceptions handling

Create and implement the following exceptions in your application.

• Invalid Pet Age Handling: o In the Pet Adoption Platform, when adding a new pet to a shelter, the age of the pet should be a positive integer. Write a program that prompts the user to input the age of a pet. Implement exception handling to ensure that the input is a positive integer. If the input is not valid, catch the exception and display an error message. If the input is valid, add the pet to the shelter.

```
class InvalidPetAge(Exception):
    pass

class NullReference(Exception):
    pass

class InsufficientFunds(Exception):
    pass

class FileHandlingError(Exception):
    pass

class AdoptionException(Exception):
    pass
```

```
try:
    def petAge():
        age = int(input("Enter pet's age "))
        if age < 0 :
            raise InvalidPetAge
except InvalidPetAge:
    print("Pet's age can't be negative")

petAge()
```