Coding Challenges: PetPals

ANUSH KUMAR

🎯 Project Objective: PetPals – The Pet Adoption Platform

The PetPals project's goal is to create and put into use a complete pet adoption platform that makes it easier for people to adopt animals from shelters and rescue groups, including dogs and cats. It functions as and online marketplace in which:

- Adopters can look through the pets that are up for adoption.
- Pets from shelters can be listed for adoption.
- Donors can assist animal welfare by making monetary or material contributions.
- Classes and objects are used in object-oriented programming (OOP) to model real-world things.
- Using unique user-defined exceptions, robust exception handling is achieved.
- Database integration for adoption events, donations, shelters, and pets' long-term storage and recovery.
- Logic is divided into DAO, entity, utility, and exception packages using a modular code structure.
- menu-driven user interaction with a primary module for smooth feature presentation and navigation.

entity/adoptionevent.py:

```
class AdoptionEvent:

    def __init__(self):
        self.participants = []


    def register_participant(self, participant):
        if isinstance(participant, IAdoptable):
            self.participants.append(participant)
            print("Participant registered successfully.")
        else:
            print("Error: Participant must implement IAdoptable.")


    def host_event(self):
        print("Hosting the adoption event...")
        for participant in self.participants:
            participant.adopt()
```

entity/ cashdonation.py

```
from datetime import datetime

class cashdonation:
    def __init__(self, donor, amount, donation_date=None):
        self._donor = donor
        self._amount = amount
        self._donation_date = donation_date if donation_date else datetime.now()
```

```
    def get_donor(self):
        return self._donor

    def get_amount(self):
        return self._amount

    def get_donation_date(self):
        return self._donation_date
```

entity/ cat.py

```
class cat:
    def __init__(self, name, age, breed, cat_color):
        self.name = name
        self.age = age
        self.breed = breed
        self.cat_color = cat_color
        self.pet_type = 'cat'

    def get_name(self):
        return self.name

    def get_age(self):
        return self.age

    def get_breed(self):
        return self.breed

    def get_cat_color(self):
        return self.cat_color

    def get_dog_breed(self):
        return None
```

```
    def get_type(self):
        return self.pet_type
```

entity/ dog.py

```
class dog:
    def __init__(self, name, age, breed, dog_breed):
        self.name = name
        self.age = age
        self.breed = breed
        self.dog_breed = dog_breed
        self.pet_type = 'dog'

    def get_name(self):
        return self.name

    def get_age(self):
        return self.age

    def get_breed(self):
        return self.breed

    def get_dog_breed(self):
        return self.dog_breed

    def get_cat_color(self):
        return None

    def get_type(self):
        return self.pet_type
```

entity/ donation.py

```
from abc import ABC,abstractmethod

class donation(ABC):
    def __init__(self, donor_name, amount):
        self._donor_name = donor_name
        self._amount = amount

    def get_donor_name(self):
        return self._donor_name

    def set_donor_name(self, donor_name):
        self._donor_name = donor_name

    def get_amount(self):
        return self._amount

    def set_amount(self, amount):
        self._amount = amount

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

entity/ iadoptable.pyentity

```
from abc import ABC, abstractmethod

class IAdoptable(ABC):
    @abstractmethod
    def adopt(self):
        pass
```

entity/ itemdonation.py

```
from entity.donation import donation

class itemdonation(donation):
    def __init__(self,donar_name,amount,itemtype):
        super().__init__(donar_name,amount)
        self._itemtype=itemtype

    def get_itemtype(self):
        return self._itemtype

    def set_itemtype(self,itemtype):
        self._itemtype=itemtype
```

entity/ pet.py

```
class pet:
    def __init__(self,name,age,breed):
        self._name=name
        self._age=age
        self._breed=breed

    def get_name(self):
        return self._name

    def get_age(self):
        return self._age
```

```
    def get_breed(self):
        return self._breed

    def set_name(self,name):
        self._name=name
    def set_age(self,age):
        self._age=age

    def set_breed(self,breed):
        self._breed=breed

    def __str__(self):
        return f"Pet(Name: {self._name}, Age: {self._age}, Breed: {self._breed})"
```

entity/ petshelter.py

```
class petshelter:
    def __init__(self):
        self.pets=[]
```

dao/ adoptioneventdao.py

```
import mysql.connector
from exceptions.AdoptionException import AdoptionException
from datetime import datetime
from util.config import connect_db

class AdoptionEventDAO:
    def __init__(self, connection):
```

```
        try:
            self.conn = connection
            self.cursor = self.conn.cursor()
        except mysql.connector.Error as e:
            raise AdoptionException(f"Database connection failed: {str(e)}")

    def host_event(self, event_name):
        try:
            event_date = datetime.now()
            query = "INSERT INTO adoption_events (event_name, event_date) VALUES (%s, %s)"
            self.cursor.execute(query, (event_name, event_date))
            self.conn.commit()
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to host adoption event: {str(e)}")

    def register_participant(self, event_id, participant_id, participant_type):
        try:
            query = "INSERT INTO event_participants (event_id, participant_id, participant_type) VALUES (%s, %s, %s)"
            self.cursor.execute(query, (event_id, participant_id, participant_type))
            self.conn.commit()
            print("Participant registered successfully.")
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to register participant: {str(e)}")
```

```
    def list_events(self):
        try:
            query = "SELECT * FROM adoption_events"
            self.cursor.execute(query)
            return self.cursor.fetchall()
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to retrieve events: {str(e)}")
```

dao/ donationdao.py

```
import mysql.connector
from exceptions.InsufficientFundsException import InsufficientFundsException
from exceptions.AdoptionException import AdoptionException


class DonationDAO:
    def __init__(self, connection):
        self.conn = connection
        self.cursor = self.conn.cursor()


    def add_cash_donation(self, donation):
        try:
            query = "INSERT INTO cash_donations (donor, amount, donation_date)
VALUES (%s, %s, %s)"
```

```
            values = (donation.get_donor(), donation.get_amount(),
donation.get_donation_date())
            self.cursor.execute(query, values)
            self.conn.commit()
            print("Cash donation recorded successfully.")
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to record donation: {str(e)}")


    def add_item_donation(self, donor_name, amount, item_type):
        query = "insert into cash_donations (donor, amount, item_type) values
(%s, %s, %s)"
        self.cursor.execute(query, (donor_name, amount, item_type))
        self.conn.commit()


    def list_all_donations(self):
        query = """SELECT donor, amount AS value, donation_date FROM
cash_donations
            UNION
            SELECT donor_name, NULL AS value, donation_date FROM
item_donation;
            """
        self.cursor.execute(query)
        return self.cursor.fetchall()
```

dao/ petdao.py

```
import mysql.connector
from exceptions.InvalidPetAgeException import  InvalidPetAgeException
from exceptions.NullReferenceException import  NullReferenceException
from util.config import connect_db


class PetDAO:
    def __init__(self, connection):
        self.conn = connection
        self.cursor = self.conn.cursor()


    def add_pet(self, pet):
        if pet.get_age() <= 0:
            raise InvalidPetAgeException("Pet age must be a positive integer.")
        query = "insert into pets (name, age, breed, type, dog_breed, cat_color) values (%s, %s, %s, %s, %s, %s)"
        values = (pet.get_name(), pet.get_age(), pet.get_breed(), pet.get_type(), pet.get_dog_breed(), pet.get_cat_color())
        self.cursor.execute(query, values)
        self.conn.commit()


    def remove_pet(self, pet_id):
        query = "delete from pets where pet_id = %s"
        self.cursor.execute(query, (pet_id,))
        self.conn.commit()
```

```
    def list_all_pets(self):
        query = "select * from pets"
        self.cursor.execute(query)
        pets = self.cursor.fetchall()
        if pets is None:
            raise NullReferenceException("No pets available.")
        return pets
```

dao/ petshelterdao.py

```
import mysql.connector
from exceptions.AdoptionException import AdoptionException

class PetShelterDAO:
    def __init__(self, connection):
        self.conn = connection
        self.cursor = self.conn.cursor()

    def register_pet_to_shelter(self, pet_id, shelter_id):
        try:
            query = "UPDATE pet_shelter SET pet_id = %s WHERE shelter_id = %s"
            self.cursor.execute(query, (pet_id, shelter_id))
            self.conn.commit()
            print("Pet registered to shelter successfully.")
```

```
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to register pet to shelter: {str(e)}")

    def remove_pet_from_shelter(self, shelter_id):
        try:
            query = "UPDATE pet_shelter SET pet_id = NULL WHERE shelter_id
= %s"
            self.cursor.execute(query, (shelter_id,))
            self.conn.commit()
        except mysql.connector.Error as e:
            raise AdoptionException(f"Failed to remove pet from shelter: {str(e)}")

    def list_all_shelters(self):
        try:
            query = "SELECT * FROM pet_shelter"
            self.cursor.execute(query)
            return self.cursor.fetchall()
        except Exception as e:
            print(f"Error retrieving shelters: {e}")
            return []
```

exception/ AdoptionException.py

```
class AdoptionException(Exception):
    def __init__(self, message="Adoption operation failed."):
```

```
        super().__init__(message)
```

exception/ FileHandlingException.py

```
class FileHandlingException(Exception):
    def __init__(self, message="Error while handling the file."):
        super().__init__(message)
```

exception/ InsufficientFundsException.py

```
class InsufficientFundsException(Exception):
    def __init__(self, message="Donation amount is less than the minimum allowed ($10)."):
        super().__init__(message)
```

exception/ InvalidPetAgeException.py

```
class InvalidPetAgeException(Exception):
    def __init__(self, message="Pet age must be a positive integer."):
        super().__init__(message)
```

exceptions/ NullReferenceException.py

```
class NullReferenceException(Exception):
    def __init__(self,message="Null reference encountered in pet details."):
        super().__init__(message)
```

util/ config.py

```
import mysql.connector


def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="root@123",
        database="petpals1"
    )
```

main/ main.py

```
from dao.petdao import PetDAO
from dao.petshelterdao import PetShelterDAO
from dao.donationdao import DonationDAO
from dao.adoptioneventdao import AdoptionEventDAO
from exceptions.InvalidPetAgeException import InvalidPetAgeException
from exceptions.NullReferenceException import NullReferenceException
from exceptions.InsufficientFundsException import InsufficientFundsException
from exceptions.FileHandlingException import FileHandlingException
from exceptions.AdoptionException import AdoptionException
from entity.dog import dog
from entity.cat import cat
from entity.cashdonation import cashdonation
```

```
from util.config import connect_db
from datetime import datetime

if __name__ == "__main__":
    connection = connect_db()
    pet_dao = PetDAO(connection)
    shelter_dao = PetShelterDAO(connection)
    donation_dao = DonationDAO(connection)
    adoption_event_dao = AdoptionEventDAO(connection)

    while True:
        print("\n=== PetPals Main Menu ===")
        print("1. Add a Pet")
        print("2. List All Pets")
        print("3. Make Cash Donation")
        print("4. Host Adoption Event")
        print("5. List All Shelters")
        print("6. Register Pet to Shelter")
        print("7. View All Donations")
        print("8. Register Participant to Event")
        print("9. List All Adoption Events")
        print("10. Exit")

        choice = input("Enter your choice: ")
```

```
try:
    if choice == "1":
        name = input("Enter pet name: ")
        age = int(input("Enter pet age: "))
        if age <= 0:
            raise InvalidPetAgeException("Pet age must be a positive integer.")
        breed = input("Enter breed: ")
        pet_type = input("Enter type (dog/cat): ").lower()
        if pet_type == "dog":
            dog_breed = input("Enter specific dog breed: ")
            pet = dog(name, age, breed, dog_breed)
        elif pet_type == "cat":
            cat_color = input("Enter cat color: ")
            pet = cat(name, age, breed, cat_color)
        pet_dao.add_pet(pet)
        print("Pet added successfully.")

    elif choice == "2":
        print("\nAll Pets:")
        pets = pet_dao.list_all_pets()
        for pet in pets:
            try:
                if pet[1] is None or pet[2] is None:
```

```
                raise NullReferenceException("Pet has missing attributes.")
            print(pet)
        except NullReferenceException as e:
            print(e)

    elif choice == "3":
        donor = input("Enter donor name: ")
        amount = float(input("Enter donation amount: "))
        if amount < 10:
            raise InsufficientFundsException("Minimum donation amount is
$10.")
        donation = cashdonation(donor, amount, datetime.now())
        donation_dao.add_cash_donation(donation)

    elif choice == "4":
        available_pets = pet_dao.list_all_pets()
        if not available_pets:
            raise AdoptionException("No pets available for adoption.")
        event_name = input("Enter the name of the adoption event: ")
        adoption_event_dao.host_event(event_name)
        print("Adoption event hosted successfully.")

    elif choice == "5":
        print("All Pet Shelters:")
        shelters = shelter_dao.list_all_shelters()
```

```
            for s in shelters:
                print(s)

        elif choice == "6":
            pet_id = int(input("Enter Pet ID: "))
            shelter_id = int(input("Enter Shelter ID: "))
            shelter_dao.register_pet_to_shelter(pet_id, shelter_id)

        elif choice == "7":
            print("All Donations:")
            donations = donation_dao.list_all_donations()
            for d in donations:
                print(d)

        elif choice == "8":
            event_id = int(input("Enter Event ID: "))
            participant_id = int(input("Enter Participant ID: "))
            participant_type = input("Enter Participant Type (pet/human): ")
            adoption_event_dao.register_participant(event_id, participant_id,
participant_type)

        elif choice == "9":
            print("All Adoption Events:")
            events = adoption_event_dao.list_events()
            for e in events:
```

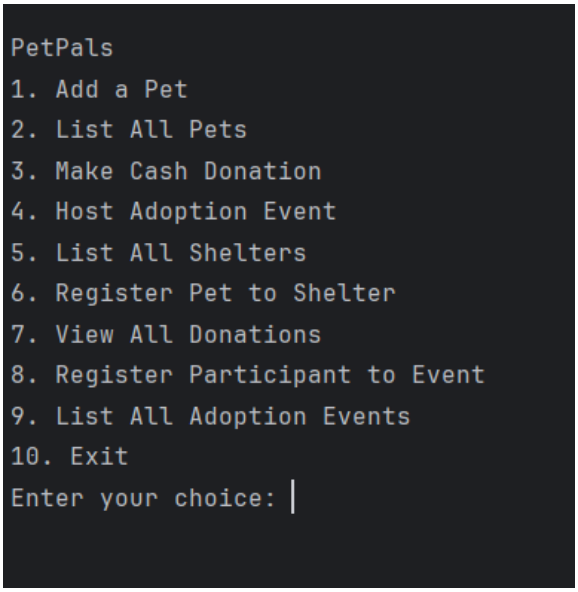
```
            print(e)

        elif choice == "10":
            print("Exiting the application.")
            break

        else:
            print("Invalid choice. Please try again.")

    except (InvalidPetAgeException, NullReferenceException,
InsufficientFundsException,
            FileHandlingException, AdoptionException, ValueError) as e:
        print(f"Error: {e}")
```

OUTPUT:



ADD PET:

```
PetPals
1. Add a Pet
2. List All Pets
3. Make Cash Donation
4. Host Adoption Event
5. List All Shelters
6. Register Pet to Shelter
7. View All Donations
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 1
Enter pet name: shadow
Enter pet age: 2
Enter breed: lab
Enter type (dog/cat): dog
Enter specific dog breed: black lab
Pet added successfully.
```

LIST ALL PETS:

```
4. Host Adoption Event
5. List All Shelters
6. Register Pet to Shelter
7. View All Donations
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 2

All Pets:
(1, 'sd', 2, None, 'dog', 'sdd', None, 'dsd')
(2, 'anush', 2, None, 'dog', 'as', None, 'lab')
(3, 'anush', 14, None, 'dog', 'as', None, 'as')
(4, 'kaml', 14, None, 'dog', 'as', None, 'as')
(5, 'ddd', 55, None, 'dog', 'as', None, 'sa')
(6, 'jj', 1, None, 'dog', 'sa', None, 'as')
(7, 'shadow', 2, None, 'dog', 'black lab', None, 'lab')
```

MAKE CASH DONATION:

```
7. View All Donations
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 4
Enter the name of the adoption event: farewell
Adoption event hosted successfully.
```

HOST ADOPTION EVENT:

```
7. View All Donations
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 4
Enter the name of the adoption event: farewell
Adoption event hosted successfully.
```

LIST ALL SHELTER:

```
7. View All Donations
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 5
All Pet Shelters:
(1, 'Twisty Tails', 1)
```

REGISTER PET TO SHELTER:

```
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 6
Enter Pet ID: 1
Enter Shelter ID: 1
Pet registered to shelter successfully.
```

VIEW ALL DONATION:

```
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 7
All Donations:
('anush', Decimal('200.00'), datetime.datetime(2025, 4, 9, 18, 25, 49))
('l', Decimal('14.00'), datetime.datetime(2025, 4, 9, 19, 17))
('anush', Decimal('100000.00'), datetime.datetime(2025, 4, 9, 21, 3, 17))
```

REGISTER PARTICIPANT TO EVENT:

```
9. List All Adoption Events
10. Exit
Enter your choice: 8
Enter Event ID: 1
Enter Participant ID: 2
Enter Participant Type (pet/human): human
Participant registered successfully.
```

LIST ALL ADOPTION EVENTS:

```
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 9
All Adoption Events:
(1, 'manga', datetime.date(2025, 4, 9))
(2, 'mako', datetime.date(2025, 4, 9))
(3, 'makal', datetime.date(2025, 4, 9))
(4, 'farewell', datetime.date(2025, 4, 9))
```

EXIT:

```
8. Register Participant to Event
9. List All Adoption Events
10. Exit
Enter your choice: 10
Exiting
```