Coding challenges :

Petpals

Project Objective: PetPals – The Pet Adoption Platform

The aim of the PetPals project is to develop a simple-to-use platform that assists individuals in adopting pets such as dogs and cats from shelters and rescue centers. It functions as an online marketplace where:

1)Individuals can look for pets up for adoption.

2)Shelters can post pets to be listed for adoption.

3)Donors can assist animals by donating money or other resources.

The project employs object-oriented programming (OOP) to represent real-world entities such as pets, shelters, and users through classes and objects. It also correctly handles errors with custom exceptions.

All the pet-related data, adoption events, donations, and shelters' data are stored in a MySQL database so that information is preserved and can be retrieved later.

The project is neatly structured with independent sections (packages) for:

-- entity for pet and shelter details,

--dao - for database operations,

--util-for utility functions,

--exception for error handling.

entity/adoptionevent.py

```
from entity.iadoptable import IAdoptable

class AdoptionEvent:
    def __init__(self):
        self._participants = []

    def register_participant(self, participant: IAdoptable):
        self._participants.append(participant)

    def host_event(self):
        for participant in self._participants:
            participant.adopt()
        return "Adoption event hosted successfully!"
```

entity/cashdonation.py

```
from entity.donation import Donation

class CashDonation(Donation):
    def __init__(self, donor_name=None, amount=None, donation_date=None):
        super().__init__(donor_name, amount)
        self._donation_date = donation_date


    def donation_date(self):
        return self._donation_date

    def donation_date(self, value):
        self._donation_date = value

    def record_donation(self):
        return f"Cash donation recorded: {self._donor_name} donated
${self._amount} on {self._donation_date}"
```

entity/cat.py

```
from entity.pet import Pet
```

```
class Cat(Pet):
    def __init__(self, name=None, age=None, breed=None, cat_color=None):
        super().__init__(name, age, breed)
        self._cat_color = cat_color

    def cat_color(self):
        return self._cat_color

    def cat_color(self, value):
        self._cat_color = value

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

entity/dog.py

```
from entity.pet import Pet

class Dog(Pet):
    def __init__(self, name=None, age=None, breed=None, dog_breed=None):
        super().__init__(name, age, breed)
        self._dog_breed = dog_breed

    def dog_breed(self):
        return self._dog_breed

    def dog_breed(self, value):
        self._dog_breed = value

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

entity/donation.py

```
from abc import ABC, abstractmethod
```

```
class Donation(ABC):
    def __init__(self, donor_name=None, amount=None):
        self._donor_name = donor_name
        self._amount = amount

    def donor_name(self):
        return self._donor_name

    def donor_name(self, value):
        self._donor_name = value
    def amount(self):
        return self._amount

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

    def record_donation(self):
        pass
```

entity/iadoptable.py

```
from abc import ABC, abstractmethod

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

entity/itemdonation.py

```
from entity.donation import Donation

class ItemDonation(Donation):
    def __init__(self, donor_name=None, amount=None, item_type=None):
        super().__init__(donor_name, amount)
        self._item_type = item_type


    def item_type(self):
```

```
        return self._item_type


    def item_type(self, value):
        self._item_type = value

    def record_donation(self):
        return f"Item donation recorded: {self._donor_name} donated
{self._item_type} worth ${self._amount}"
```

entity/pet.py

```
class Pet:
    def __init__(self, name=None, age=None, breed=None):
        self._name = name
        self._age = age
        self._breed = breed


   def name(self):
        return self._name
    def name(self, value):
        self._name = value

    def age(self):
        return self._age

    def age(self, value):
        self._age = value

       def breed(self):
        return self._breed

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

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

entity/petshelter.py

```
from entity.pet import Pet

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

    def add_pet(self, pet: Pet):
        self._available_pets.append(pet)

    def remove_pet(self, pet: Pet):
        if pet in self._available_pets:
            self._available_pets.remove(pet)
        else:
            raise Exception("Pet not found in shelter")

    def list_available_pets(self):
        return self._available_pets
```

dao/ipetrepository.py

```
from abc import ABC, abstractmethod
from entity.pet import Pet
from entity.donation import Donation

class IPetRepository(ABC):

    def add_pet(self, pet: Pet) -> bool:
        pass


    def list_pets(self) -> list:
        pass

    def record_donation(self, donation: Donation) -> bool:
        pass

    def register_event_participant(self, event_id: int, participant_type: str) ->
```

```
bool:
        pass
```

dao/petrepositoryimpl.py

```
import mysql.connector
from dao.ipet_repository import IPetRepository
from entity.pet import Pet
from entity.dog import Dog
from entity.cat import Cat  # Also needed for Cat check
from entity.cash_donation import CashDonation
from entity.donation import Donation
from entity.item_donation import ItemDonation
from util.db_conn_util import DBConnUtil

class PetRepositoryImpl(IPetRepository):
    def __init__(self):
        self.connection = DBConnUtil.get_connection()

    def add_pet(self, pet: Pet) -> bool:
        cursor = self.connection.cursor()
        if isinstance(pet, Dog):
            query = "INSERT INTO Pets (name, age, breed, type, dog_breed) VALUES
(%s, %s, %s, 'Dog', %s)"
            cursor.execute(query, (pet.name, pet.age, pet.breed, pet.dog_breed))
        elif isinstance(pet, Cat):
            query = "INSERT INTO Pets (name, age, breed, type, cat_color) VALUES
(%s, %s, %s, 'Cat', %s)"
            cursor.execute(query, (pet.name, pet.age, pet.breed, pet.cat_color))
        self.connection.commit()
        return True

    def list_pets(self) -> list:
        cursor = self.connection.cursor()
        cursor.execute("SELECT * FROM Pets")
        rows = cursor.fetchall()
```

```
        pets = []
        for row in rows:
            if row[4] == 'Dog':
                pet = Dog(row[1], row[2], row[3], row[5])
            elif row[4] == 'Cat':
                pet = Cat(row[1], row[2], row[3], row[6])
            pet.pet_id = row[0]
            pets.append(pet)
        return pets

    def record_donation(self, donation: Donation) -> bool:
        cursor = self.connection.cursor()
        if isinstance(donation, CashDonation):
            query = "INSERT INTO Donations (donor_name, amount, donation_type,
donation_date) VALUES (%s, %s, 'Cash', %s)"
            cursor.execute(query, (donation.donor_name, donation.amount,
donation.donation_date))
        elif isinstance(donation, ItemDonation):
            query = "INSERT INTO Donations (donor_name, amount, donation_type,
item_type) VALUES (%s, %s, 'Item', %s)"
            cursor.execute(query, (donation.donor_name, donation.amount,
donation.item_type))
        self.connection.commit()
        return True

    def register_event_participant(self, event_id: int, participant_type: str) ->
bool:
        cursor = self.connection.cursor()
        query = "INSERT INTO Participants (event_id, participant_type) VALUES
(%s, %s)"
        cursor.execute(query, (event_id, participant_type))
        self.connection.commit()
        return True
```

exception/exceptions.py

```
class InvalidPetAgeException(Exception):
    pass
```

```
class NullReferenceException(Exception):
    pass

class InsufficientFundsException(Exception):
    pass

class FileHandlingException(Exception):
    pass

class AdoptionException(Exception):
    pass
```

util/dbconnutil.py

```
import mysql.connector
from util.db_property_util import DBPropertyUtil

class DBConnUtil:
    connection = None

    def get_connection():
        if DBConnUtil.connection is None:
            conn_string = DBPropertyUtil.get_property_string("db.properties")
            config = {
                'host': conn_string.split('@')[1].split(':')[0],
                'user': conn_string.split('://')[1].split(':')[0],
                'password': conn_string.split(':')[2].split('@')[0],
                'database': conn_string.split('/')[3],
                'port': int(conn_string.split(':')[3].split('/')[0])
            }
            DBConnUtil.connection = mysql.connector.connect(**config)
        return DBConnUtil.connection
```

util/dbpropertyutil.py

```
import configparser

class DBPropertyUtil:

    def get_property_string(filename: str) -> str:
        config = configparser.ConfigParser()
        config.read(filename)
        db_config = config['DATABASE']
        return
f"mysql+mysqlconnector://{db_config['username']}:{db_config['password']}@{
db_config['hostname']}:{db_config['port']}/{db_config['dbname']}"
```

main/mainmodule.py

```
from dao.pet_repository_impl import PetRepositoryImpl
from entity.dog import Dog
from entity.cat import Cat
from entity.cash_donation import CashDonation
from entity.item_donation import ItemDonation
from exception.exceptions import *
from entity.pet_shelter import PetShelter

class MainModule:
    def __init__(self):
        self.repo = PetRepositoryImpl()
        self.shelter = PetShelter()

    def menu(self):
        while True:
            print("\n=== PetPals: The Pet Adoption Platform ===")
            print("1. Add Pet")
            print("2. List Available Pets")
            print("3. Record Cash Donation")
            print("4. Record Item Donation")
            print("5. Register for Adoption Event")
            print("6. 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 positive")
                breed = input("Enter pet breed: ")
                pet_type = input("Enter pet type (Dog/Cat): ").lower()
                if pet_type == "dog":
                    dog_breed = input("Enter 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)
                else:
                    raise AdoptionException("Invalid pet type")
                self.repo.add_pet(pet)
                self.shelter.add_pet(pet)
                print("Pet added successfully!")

            elif choice == "2":
                pets = self.repo.list_pets()
                if not pets:
                    raise NullReferenceException("No pets available")
                for pet in pets:
                    print(pet)

            elif choice == "3":
                donor_name = input("Enter donor name: ")
                amount = float(input("Enter donation amount: "))
                if amount < 10:
                    raise InsufficientFundsException("Donation amount must be at
least $10")
                donation_date = input("Enter donation date (YYYY-MM-DD): ")
                donation = CashDonation(donor_name, amount, donation_date)
                self.repo.record_donation(donation)
                print(donation.record_donation())
```

```
            elif choice == "4":
                donor_name = input("Enter donor name: ")
                amount = float(input("Enter donation value: "))
                item_type = input("Enter item type: ")
                donation = ItemDonation(donor_name, amount, item_type)
                self.repo.record_donation(donation)
                print(donation.record_donation())

            elif choice == "5":
                event_id = int(input("Enter event ID: "))
                self.repo.register_event_participant(event_id, "Shelter")
                print("Registered for adoption event!")

            elif choice == "6":
                print("Exiting...")
                break

            else:
                print("Invalid choice!")

        except InvalidPetAgeException as e:
            print(f"Error: {e}")
        except NullReferenceException as e:
            print(f"Error: {e}")
        except InsufficientFundsException as e:
            print(f"Error: {e}")
        except AdoptionException as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"An error occurred: {e}")

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

ADDING PET FOR ADOPTION

```
=== PetPals: The Pet Adoption Platform ===
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Record Item Donation
5. Register for Adoption Event
6. Exit
Enter your choice: 1
Enter pet name: kity
Enter pet age: 2
Enter pet breed: german
Enter pet type (Dog/Cat): dog
Enter dog breed: german shepred
Pet added successfully!
```

LIST AVAILABLE PETS

```
=== PetPals: The Pet Adoption Platform ===
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Record Item Donation
5. Register for Adoption Event
6. Exit
Enter your choice: 2
Dog(Name: a, Age: 12, Breed: b, DogBreed: vb)
Dog(Name: kity, Age: 2, Breed: german, DogBreed: german shepred)
Cat(Name: mini, Age: 3, Breed: mmaine, CatColor: brown)
Dog(Name: tonny, Age: 6, Breed: golden retriever, DogBreed: golden)
Dog(Name: diva, Age: 2, Breed: german, DogBreed: german shepred)
```

RECORD CASH DONATIONS

```
=== PetPals: The Pet Adoption Platform ===
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Record Item Donation
5. Register for Adoption Event
6. Exit
Enter your choice: 3
Enter donor name: rahul
Enter donation amount: 5000
Enter donation date (YYYY-MM-DD): 2025-04-09
Cash donation recorded: rahul donated $5000.0 on 2025-04-09
```

RECORD ITEM DONATION

```
=== PetPals: The Pet Adoption Platform ===
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Record Item Donation
5. Register for Adoption Event
6. Exit
Enter your choice: 4
Enter donor name: keerthi
Enter donation value: 3300
Enter item type: food
Item donation recorded: keerthi donated food worth $3300.0
```

EXITING

```
=== PetPals: The Pet Adoption Platform ===
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Record Item Donation
5. Register for Adoption Event
6. Exit
Enter your choice: 6
Exiting...
```