PETPALS

(CODE CHALLENGE)

Coding Challenges: PetPals, The Pet Adoption Platform

Instructions

- Project submissions should be done through the partcipants' Github repository, and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive application implemented with a strong focus on SQL, control flow statements, loops, arrays, collections, exception handling, database interaction.
- Follow object-oriented principles throughout the project. Use classes and objects to model real world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exceptions from corresponding methods and handled.
- The following Directory structure is to be followed in the application.
    - **entity/model**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider Interface/Abstract Class to showcase functionalities.
        - Create the implementation class for the above Interface/Abstract Class with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
        - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object (Use method defined in DBPropertyUtil class to get the connection String).
    - **Main**
        - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Problem Statement:

PetPals, The Pet Adoption Platform scenario is a software system designed to facilitate the adoption of pets, such as dogs and cats, from shelters or rescue organizations. This platform serves as a digital marketplace where potential adopters can browse and select pets, shelters can list available pets, and donors can contribute to support animal welfare.

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.

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.

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.

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.

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).

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.

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.

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.

6.Exceptions handling

Create and implement the following exceptions in your application.

- Invalid Pet Age Handling:
  - 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.
- Null Reference Exception Handling:
  - In the Pet Adoption Platform, when displaying the list of available pets in a shelter, it's important to handle situations where a pet's properties (e.g., Name, Age) might be null. Implement exception handling to catch null reference exceptions when accessing properties of pets in the shelter and display a message indicating that the information is missing.
- Insufficient Funds Exception:
  - Suppose the Pet Adoption Platform allows users to make cash donations to shelters. Write a program that prompts the user to enter the donation amount. Implement exception handling to catch situations where the donation amount is less than a minimum allowed amount (e.g., $10). If the donation amount is insufficient, catch the exception and display an error message. Otherwise, process the donation.
- File Handling Exception:
  - In the Pet Adoption Platform, there might be scenarios where the program needs to read data from a file (e.g., a list of pets in a shelter). Write a program that attempts to read data from a file. Implement exception handling to catch any file-related exceptions (e.g., FileNotFoundException) and display an error message if the file is not found or cannot be read.
- Custom Exception for Adoption Errors:

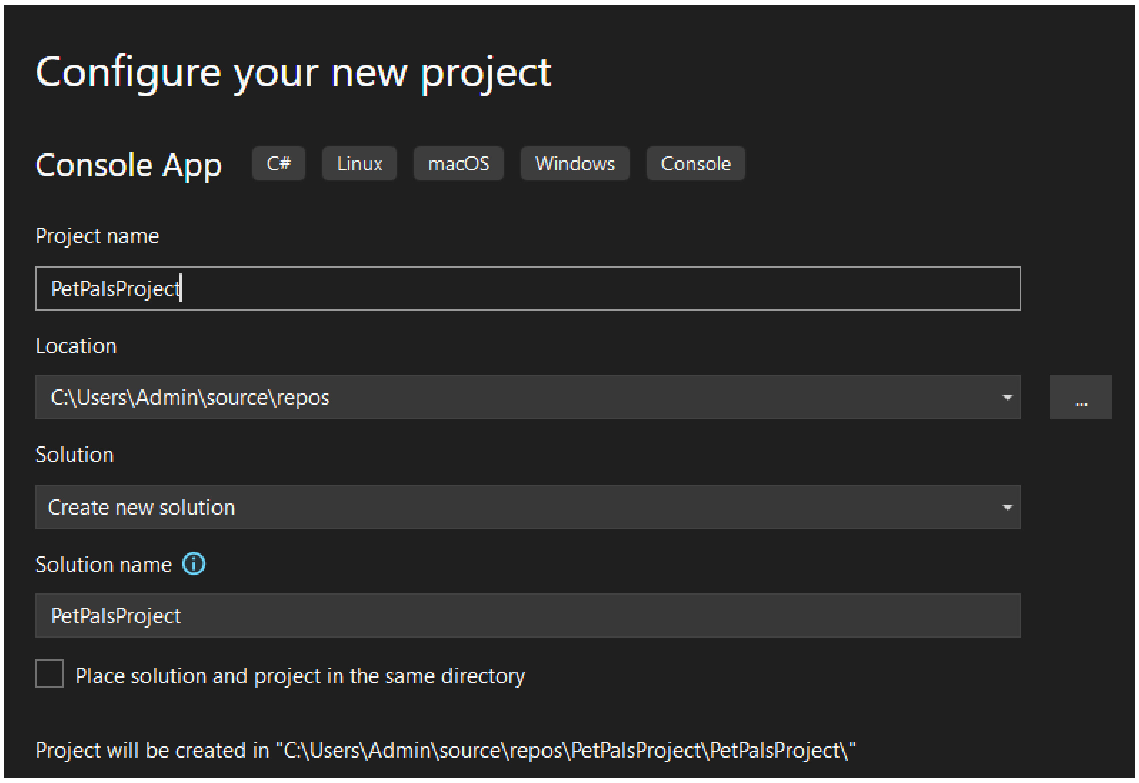
- Design a custom exception class called AdoptionException that inherits from Exception. In the Pet Adoption Platform, use this custom exception to handle adoption-related errors, such as attempting to adopt a pet that is not available or adopting a pet with missing information. Create instances of AdoptionException with different error messages and catch them appropriately in your program.

7.Database Connectivity

Create and implement the following tasks in your application.

- Displaying Pet Listings:
  - 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.
- Donation Recording:
  - 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.
- Adoption Event Management:
  - 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.

Project Setup in Visual Studio 2022



Configure your new project

Console App

Project name: PetPalsProject

Location: C:\Users\Admin\source\repos

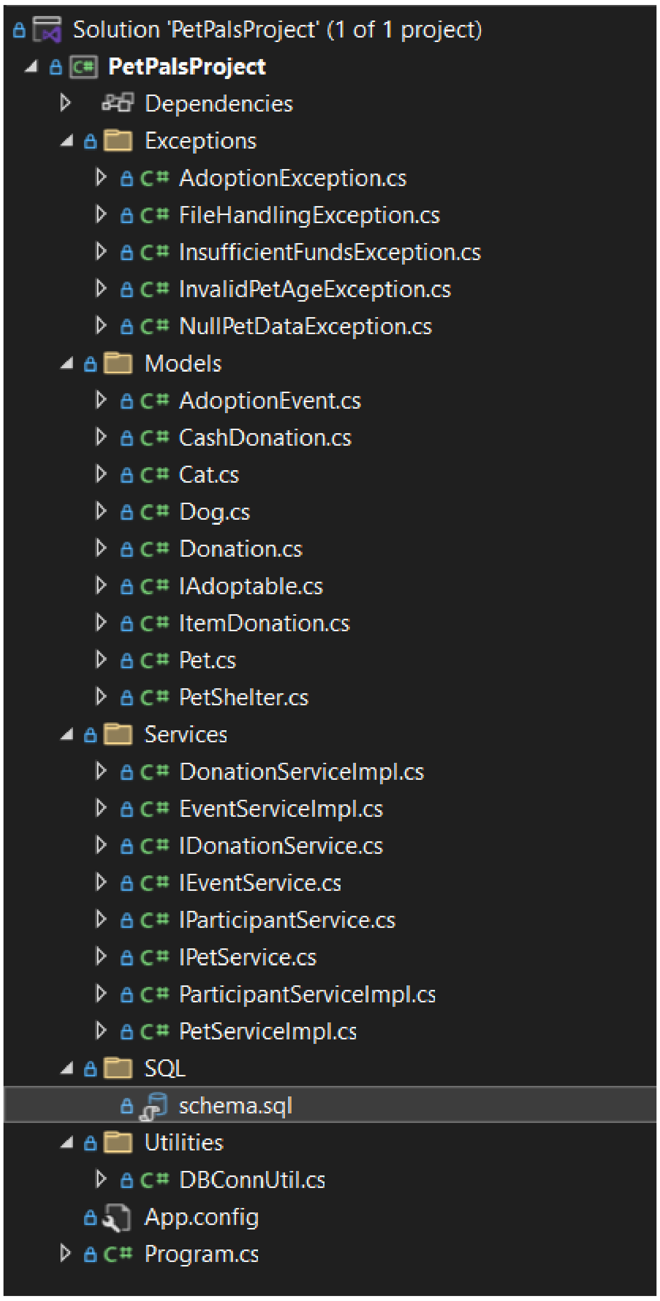
Solution: Create new solution

Solution name: PetPalsProject

☐ Place solution and project in the same directory

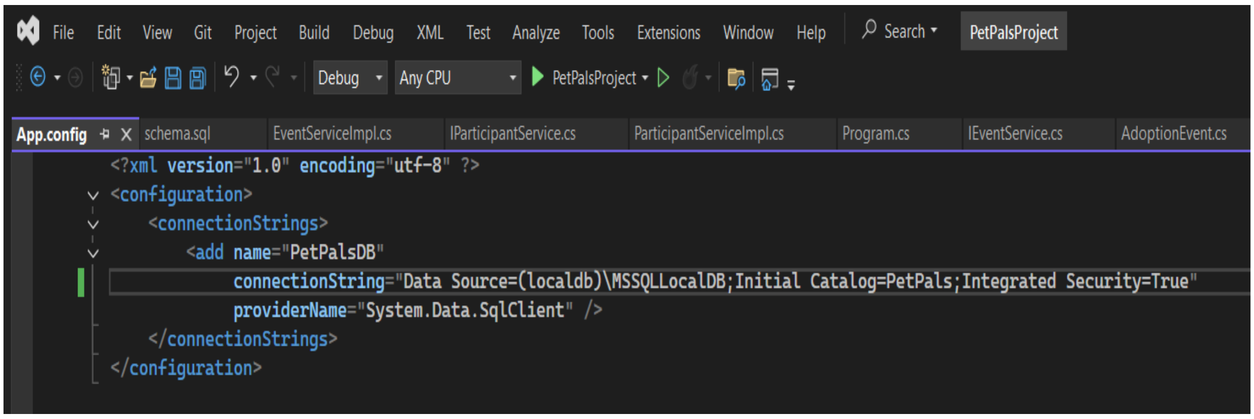
Project will be created in "C:\Users\Admin\source\repos\PetPalsProject\PetPalsProject\"

Folder Structure



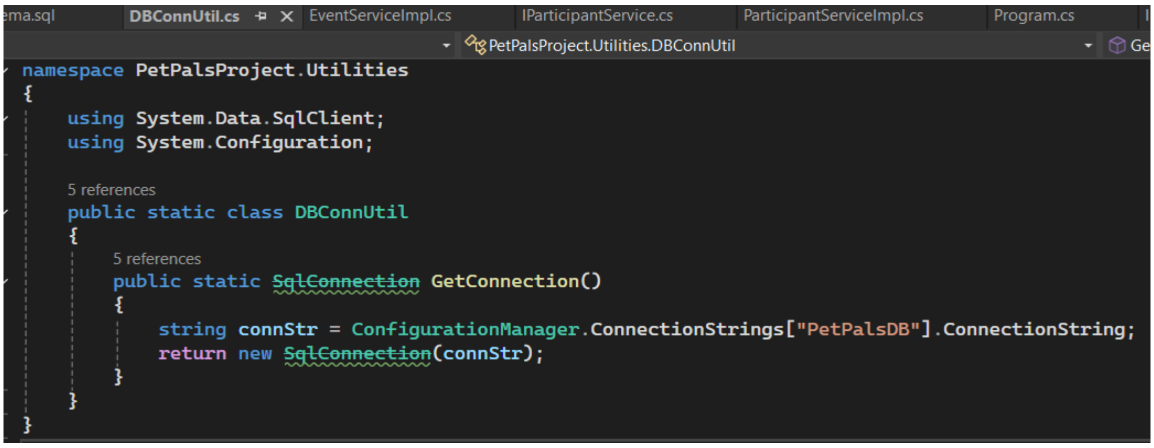
```
Solution 'PetPalsProject' (1 of 1 project)
  PetPalsProject
    Dependencies
    Exceptions
      AdoptionException.cs
      FileHandlingException.cs
      InsufficientFundsException.cs
      InvalidPetAgeException.cs
      NullPetDataException.cs
    Models
      AdoptionEvent.cs
      CashDonation.cs
      Cat.cs
      Dog.cs
      Donation.cs
      IAdoptable.cs
      ItemDonation.cs
      Pet.cs
      PetShelter.cs
    Services
      DonationServiceImpl.cs
      EventServiceImpl.cs
      IDonationService.cs
      IEventService.cs
      IParticipantService.cs
      IPetService.cs
      ParticipantServiceImpl.cs
      PetServiceImpl.cs
    SQL
      schema.sql
    Utilities
      DBConnUtil.cs
    App.config
    Program.cs
```

App.config



```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <connectionStrings>
        <add name="PetPalsDB"
             connectionString="Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=PetPals;Integrated Security=True"
             providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

DBConnUtil.cs

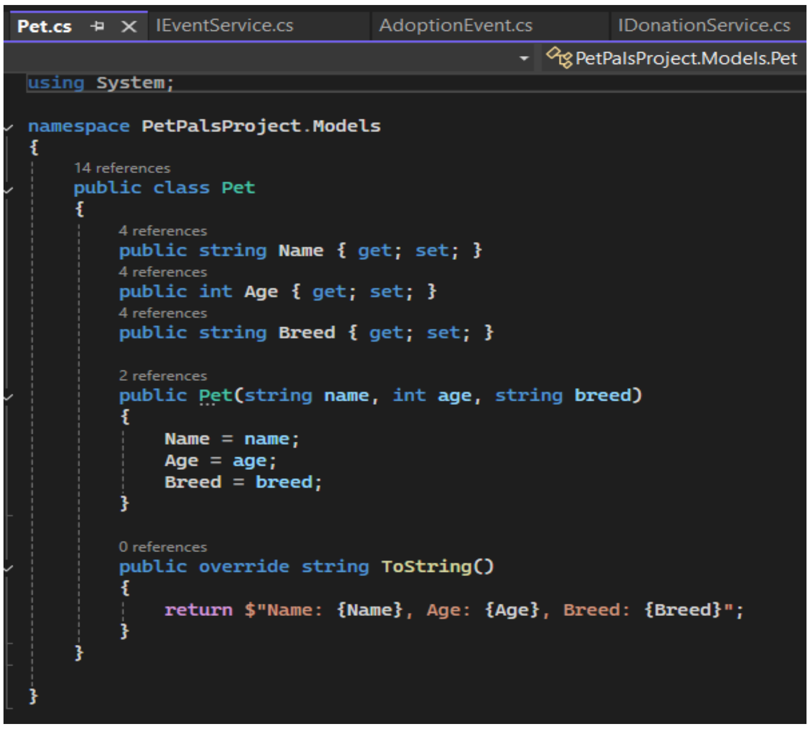


ema.sql | DBConnUtil.cs | EventServiceImpl.cs | IParticipantService.cs | ParticipantServiceImpl.cs | Program.cs

PetPalsProject.Utilities.DBConnUtil

```
namespace PetPalsProject.Utilities
{
    using System.Data.SqlClient;
    using System.Configuration;

    5 references
    public static class DBConnUtil
    {
        5 references
        public static SqlConnection GetConnection()
        {
            string connStr = ConfigurationManager.ConnectionStrings["PetPalsDB"].ConnectionString;
            return new SqlConnection(connStr);
        }
    }
}
```

Pet.cs – Base Class

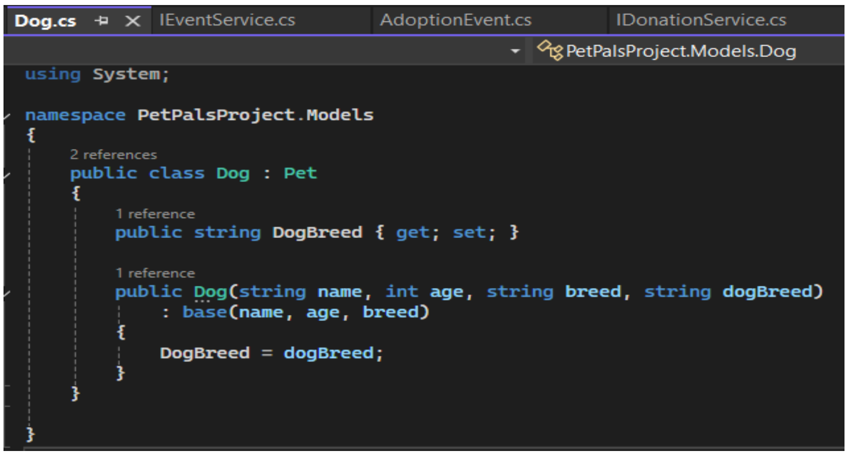


Pet.cs | IEventService.cs | AdoptionEvent.cs | IDonationService.cs

PetPalsProject.Models.Pet

```
using System;

namespace PetPalsProject.Models
{
    14 references
    public class Pet
    {
        4 references
        public string Name { get; set; }
        4 references
        public int Age { get; set; }
        4 references
        public string Breed { get; set; }

        2 references
        public Pet(string name, int age, string breed)
        {
            Name = name;
            Age = age;
            Breed = breed;
        }

        0 references
        public override string ToString()
        {
            return $"Name: {Name}, Age: {Age}, Breed: {Breed}";
        }
    }

}
```

Dog.cs – Inherits from Pet

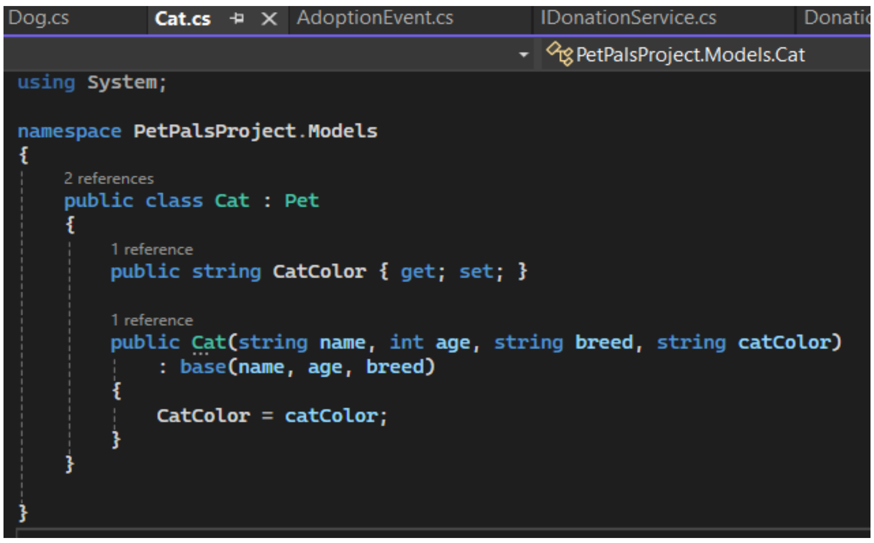


Dog.cs | IEventService.cs | AdoptionEvent.cs | IDonationService.cs

PetPalsProject.Models.Dog

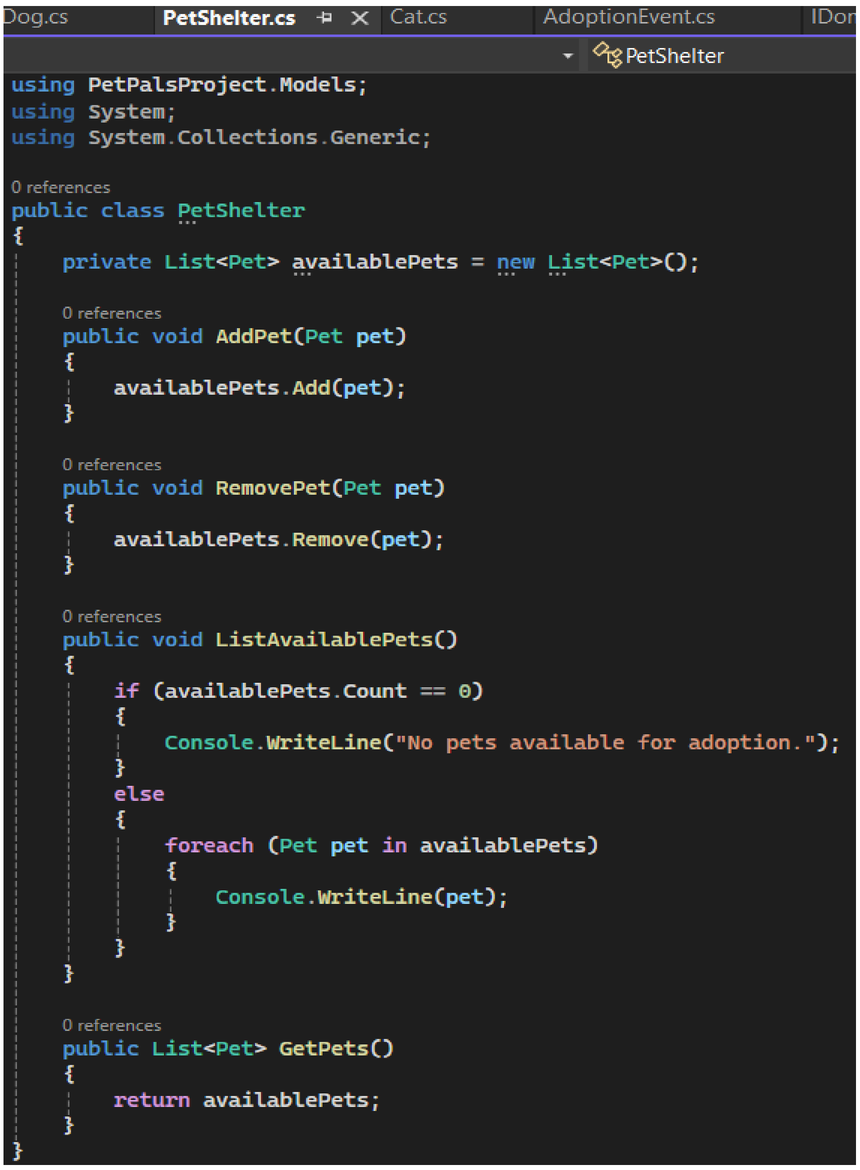
```
using System;

namespace PetPalsProject.Models
{
    2 references
    public class Dog : Pet
    {
        1 reference
        public string DogBreed { get; set; }

        1 reference
        public Dog(string name, int age, string breed, string dogBreed)
            : base(name, age, breed)
        {
            DogBreed = dogBreed;
        }
    }

}
```

Cat.cs – Inherits from Pet



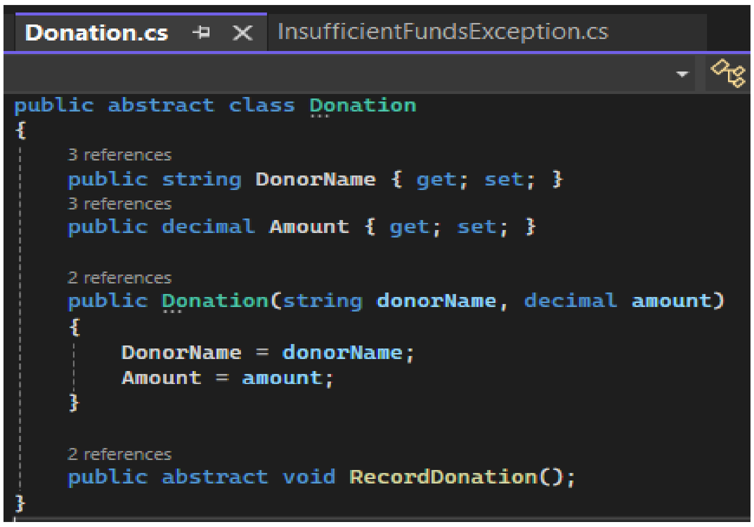
```
using System;

namespace PetPalsProject.Models
{
    2 references
    public class Cat : Pet
    {
        1 reference
        public string CatColor { get; set; }

        1 reference
        public Cat(string name, int age, string breed, string catColor)
            : base(name, age, breed)
        {
            CatColor = catColor;
        }
    }

}
```

PetShelter.cs – Manages List of Pets



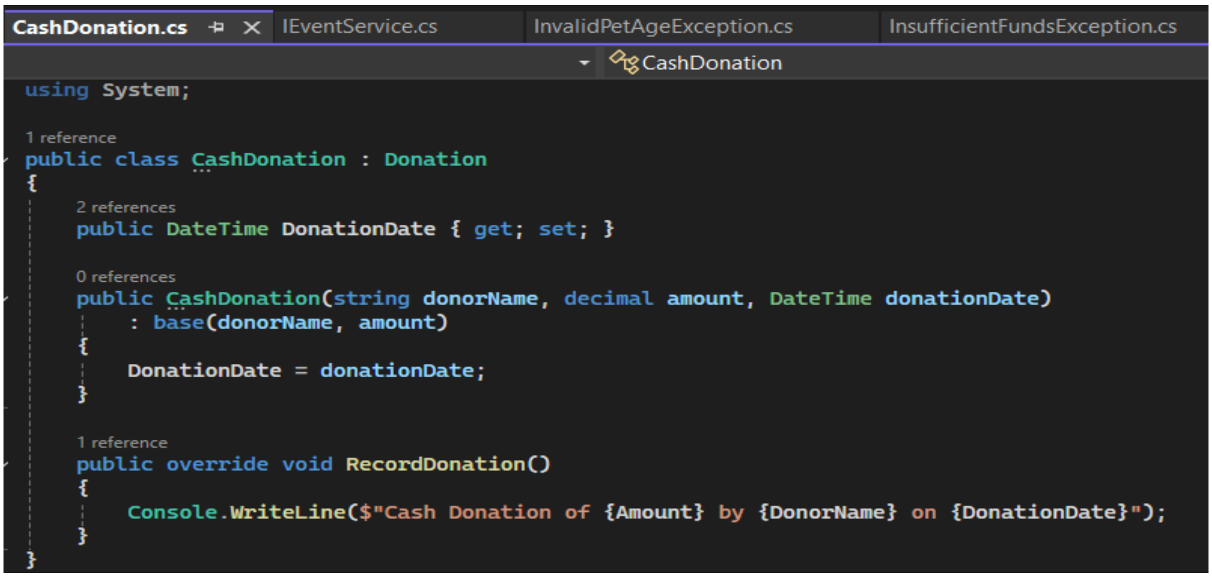
```
using PetPalsProject.Models;
using System;
using System.Collections.Generic;

0 references
public class PetShelter
{
    private List<Pet> availablePets = new List<Pet>();

    0 references
    public void AddPet(Pet pet)
    {
        availablePets.Add(pet);
    }

    0 references
    public void RemovePet(Pet pet)
    {
        availablePets.Remove(pet);
    }

    0 references
    public void ListAvailablePets()
    {
        if (availablePets.Count == 0)
        {
            Console.WriteLine("No pets available for adoption.");
        }
        else
        {
            foreach (Pet pet in availablePets)
            {
                Console.WriteLine(pet);
            }
        }
    }

    0 references
    public List<Pet> GetPets()
    {
        return availablePets;
    }
}
```

Donation.cs – Abstract Class



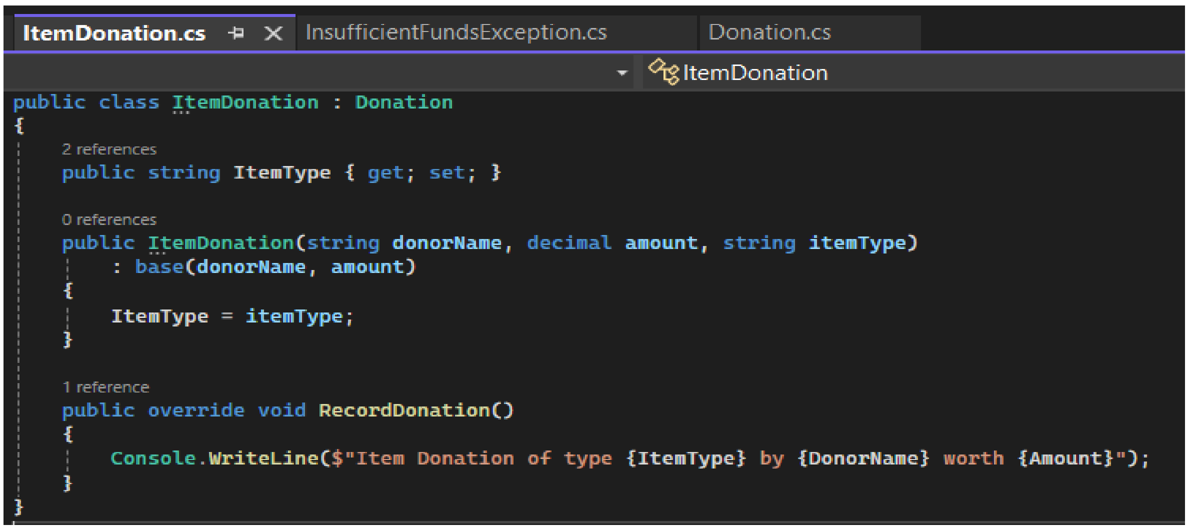
```
public abstract class Donation
{
    3 references
    public string DonorName { get; set; }
    3 references
    public decimal Amount { get; set; }

    2 references
    public Donation(string donorName, decimal amount)
    {
        DonorName = donorName;
        Amount = amount;
    }

    2 references
    public abstract void RecordDonation();
}
```

CashDonation.cs – Extends Donation



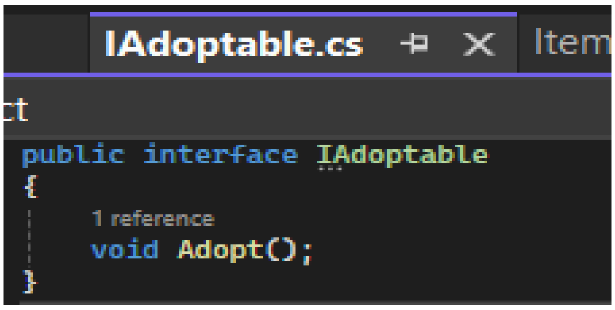
```
using System;

1 reference
public class CashDonation : Donation
{
    2 references
    public DateTime DonationDate { get; set; }

    0 references
    public CashDonation(string donorName, decimal amount, DateTime donationDate)
        : base(donorName, amount)
    {
        DonationDate = donationDate;
    }

    1 reference
    public override void RecordDonation()
    {
        Console.WriteLine($"Cash Donation of {Amount} by {DonorName} on {DonationDate}");
    }
}
```

ItemDonation.cs – Extends Donation



```
public class ItemDonation : Donation
{
    2 references
    public string ItemType { get; set; }

    0 references
    public ItemDonation(string donorName, decimal amount, string itemType)
        : base(donorName, amount)
    {
        ItemType = itemType;
    }

    1 reference
    public override void RecordDonation()
    {
        Console.WriteLine($"Item Donation of type {ItemType} by {DonorName} worth {Amount}");
    }
}
```

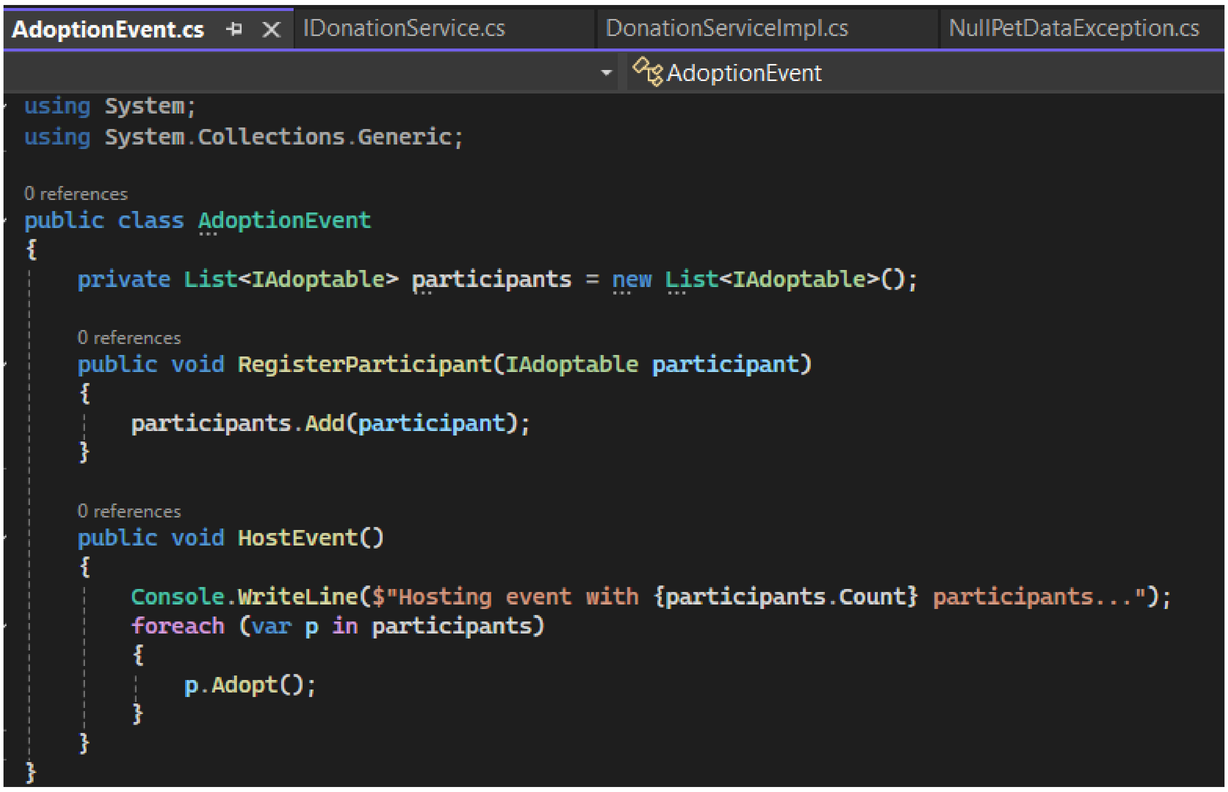
IAdoptable.cs – Interface



IAdoptable.cs
Item

```
public interface IAdoptable
{
    1 reference
    void Adopt();
}
```

AdoptionEvent.cs – Manages Participants

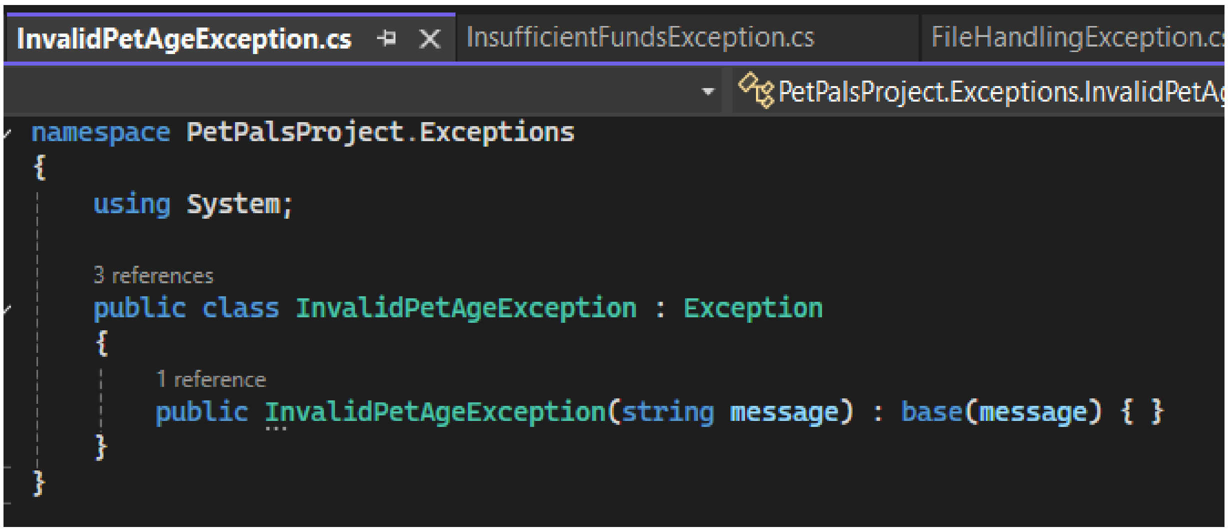


AdoptionEvent.cs | IDonationService.cs | DonationServiceImpl.cs | NullPetDataException.cs

AdoptionEvent

```
using System;
using System.Collections.Generic;

0 references
public class AdoptionEvent
{
    private List<IAdoptable> participants = new List<IAdoptable>();

    0 references
    public void RegisterParticipant(IAdoptable participant)
    {
        participants.Add(participant);
    }

    0 references
    public void HostEvent()
    {
        Console.WriteLine($"Hosting event with {participants.Count} participants...");
        foreach (var p in participants)
        {
            p.Adopt();
        }
    }
}
```

InvalidPetAgeException.cs

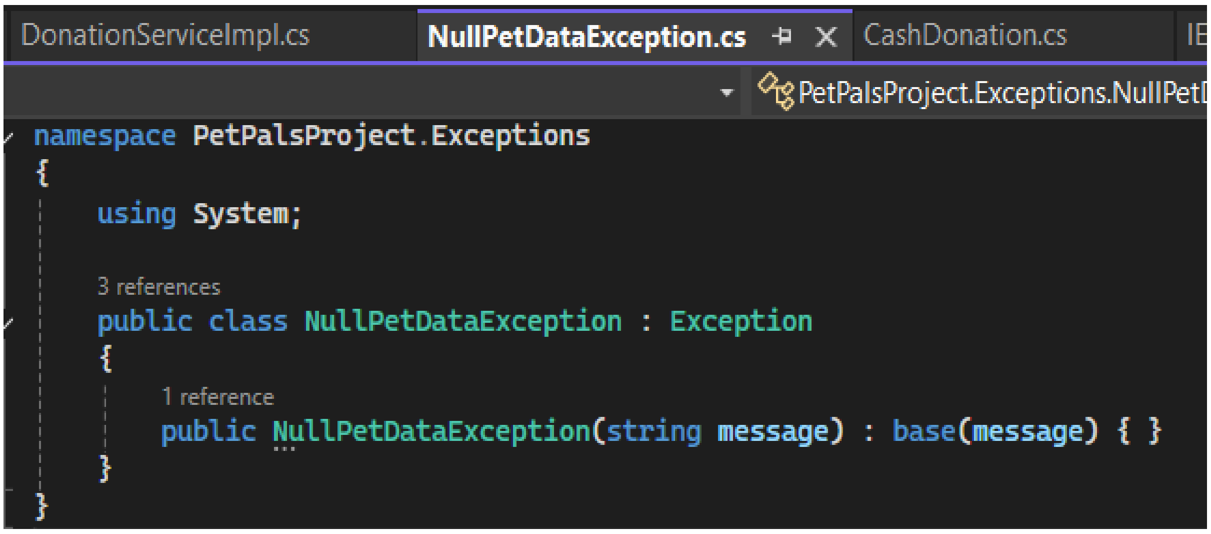


InvalidPetAgeException.cs | InsufficientFundsException.cs | FileHandlingException.c

PetPalsProject.Exceptions.InvalidPetA

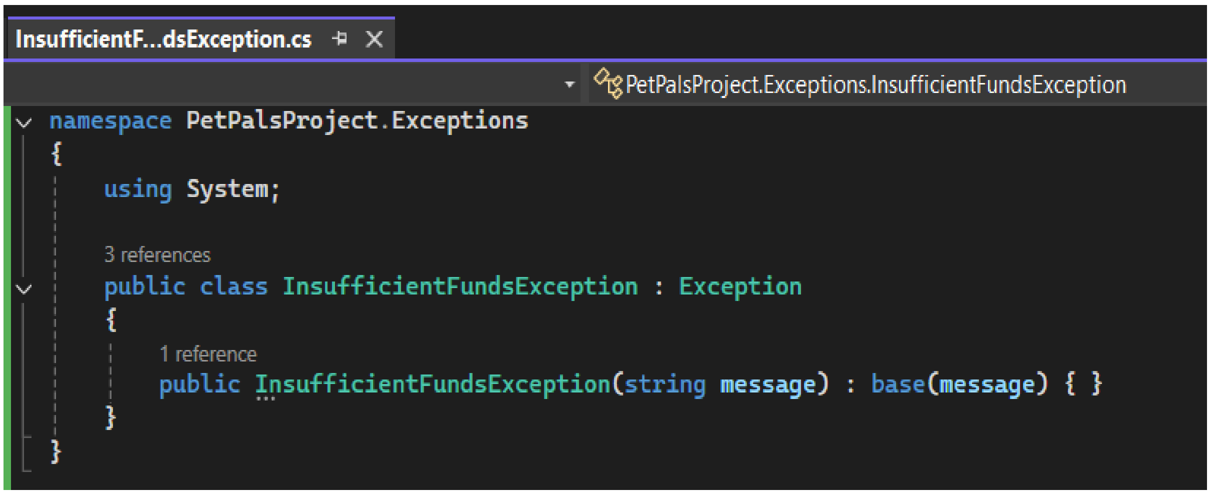
```
namespace PetPalsProject.Exceptions
{
    using System;

    3 references
    public class InvalidPetAgeException : Exception
    {
        1 reference
        public InvalidPetAgeException(string message) : base(message) { }
    }
}
```

NullPetDataException.cs



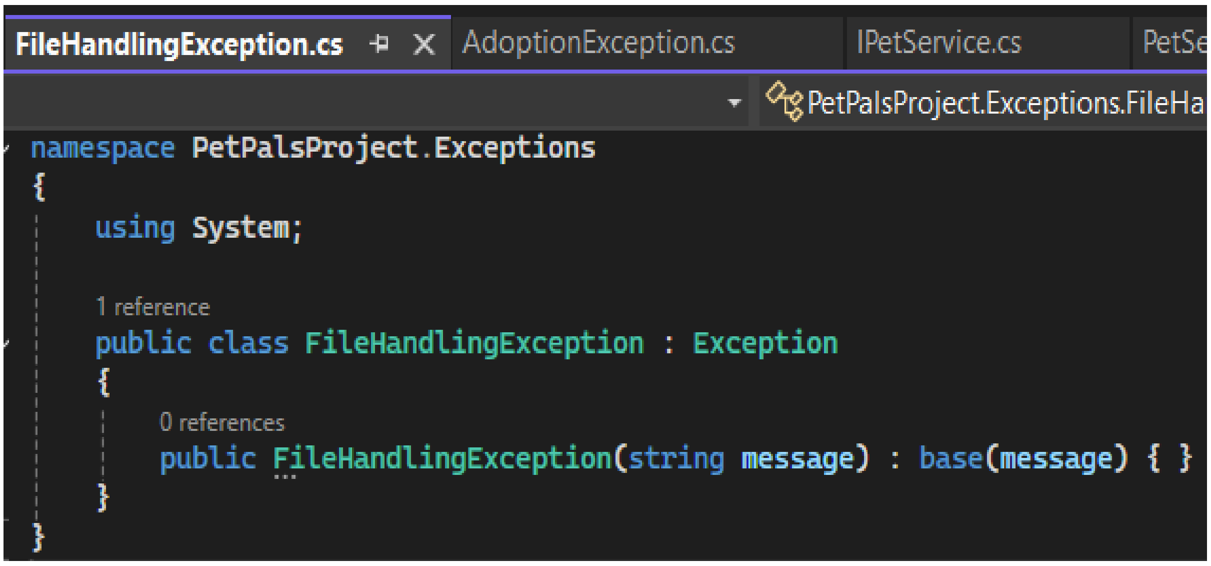
```
namespace PetPalsProject.Exceptions
{
    using System;

    3 references
    public class NullPetDataException : Exception
    {
        1 reference
        public NullPetDataException(string message) : base(message) { }
    }
}
```

InsufficientFundsException.cs


InsufficientF...dsException.cs
PetPalsProject.Exceptions.InsufficientFundsException

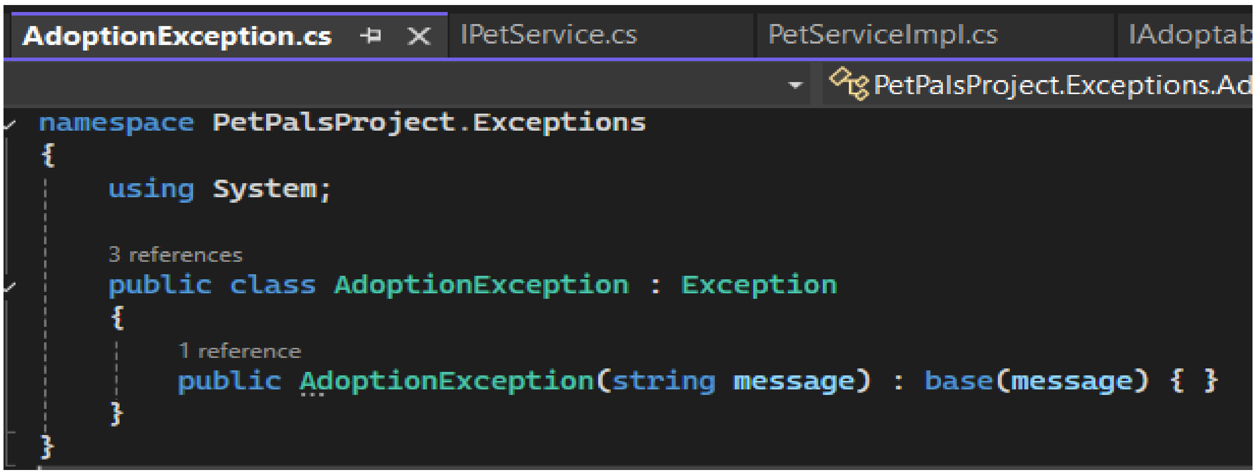
```
namespace PetPalsProject.Exceptions
{
    using System;

    3 references
    public class InsufficientFundsException : Exception
    {
        1 reference
        public InsufficientFundsException(string message) : base(message) { }
    }
}
```

FileHandlingException.cs



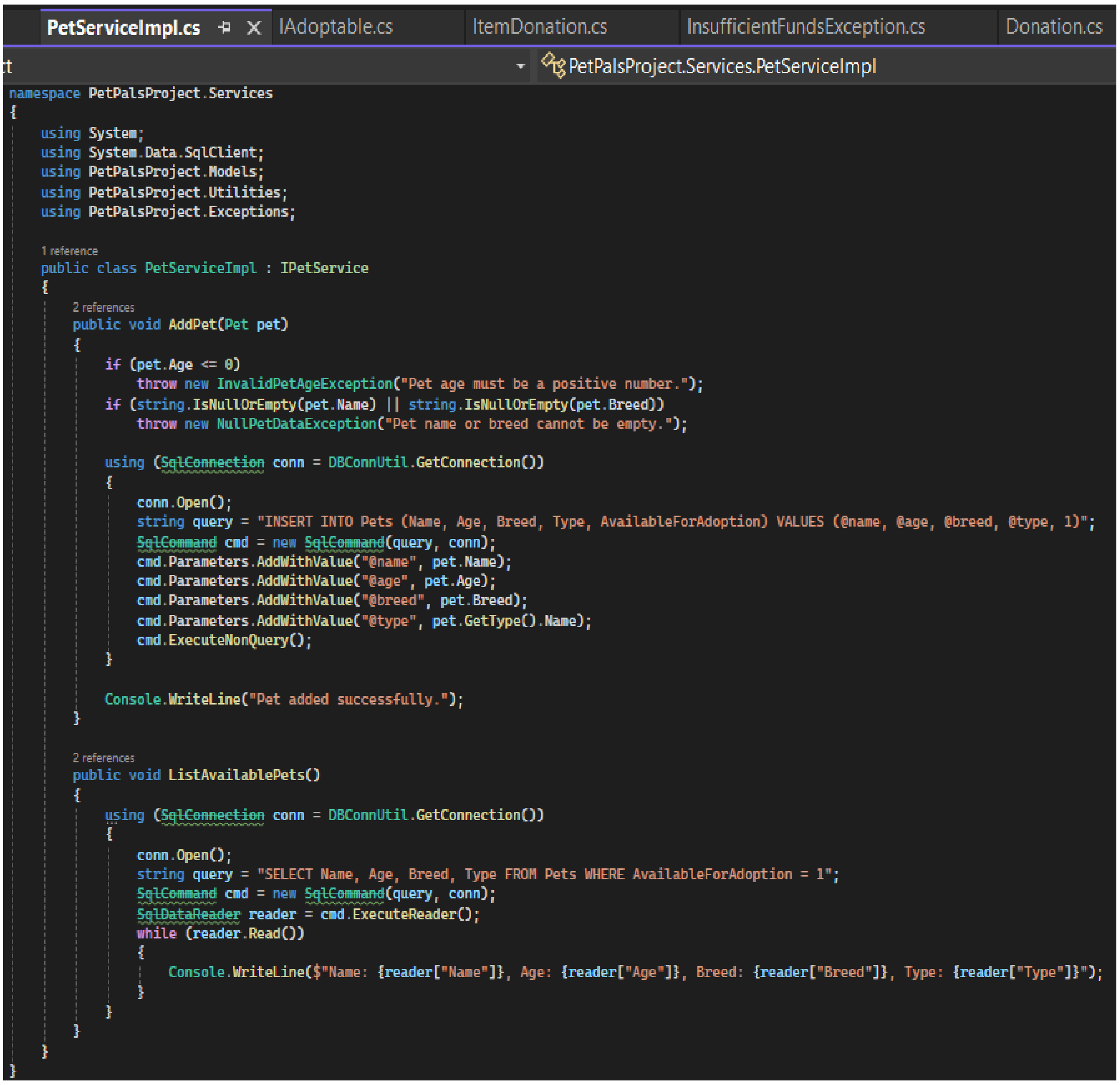
```
namespace PetPalsProject.Exceptions
{
    using System;

    1 reference
    public class FileHandlingException : Exception
    {
        0 references
        public FileHandlingException(string message) : base(message) { }
    }
}
```

AdoptionException.cs



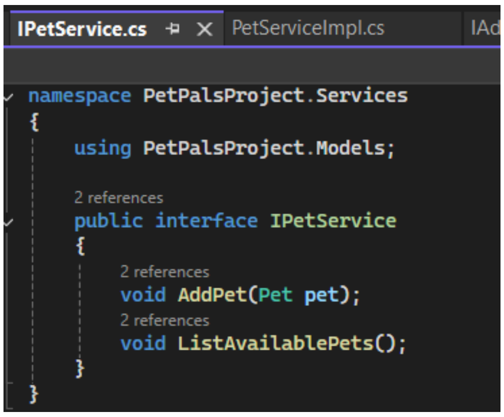
```
namespace PetPalsProject.Exceptions
{
    using System;

    3 references
    public class AdoptionException : Exception
    {
        1 reference
        public AdoptionException(string message) : base(message) { }
    }
}
```

PetServiceImpl.cs – Implements Pet Logic


PetServiceImpl.cs
IAdoptable.cs
ItemDonation.cs
InsufficientFundsException.cs
Donation.cs
PetPalsProject.Services.PetServiceImpl

```
namespace PetPalsProject.Services
{
    using System;
    using System.Data.SqlClient;
    using PetPalsProject.Models;
    using PetPalsProject.Utilities;
    using PetPalsProject.Exceptions;

    1 reference
    public class PetServiceImpl : IPetService
    {
        2 references
        public void AddPet(Pet pet)
        {
            if (pet.Age <= 0)
                throw new InvalidPetAgeException("Pet age must be a positive number.");
            if (string.IsNullOrEmpty(pet.Name) || string.IsNullOrEmpty(pet.Breed))
                throw new NullPetDataException("Pet name or breed cannot be empty.");

            using (SqlConnection conn = DBConnUtil.GetConnection())
            {
                conn.Open();
                string query = "INSERT INTO Pets (Name, Age, Breed, Type, AvailableForAdoption) VALUES (@name, @age, @breed, @type, 1)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@name", pet.Name);
                cmd.Parameters.AddWithValue("@age", pet.Age);
                cmd.Parameters.AddWithValue("@breed", pet.Breed);
                cmd.Parameters.AddWithValue("@type", pet.GetType().Name);
                cmd.ExecuteNonQuery();
            }

            Console.WriteLine("Pet added successfully.");
        }

        2 references
        public void ListAvailablePets()
        {
            using (SqlConnection conn = DBConnUtil.GetConnection())
            {
                conn.Open();
                string query = "SELECT Name, Age, Breed, Type FROM Pets WHERE AvailableForAdoption = 1";
                SqlCommand cmd = new SqlCommand(query, conn);
                SqlDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    Console.WriteLine($"Name: {reader["Name"]}, Age: {reader["Age"]}, Breed: {reader["Breed"]}, Type: {reader["Type"]}");
                }
            }
        }
    }
}
```

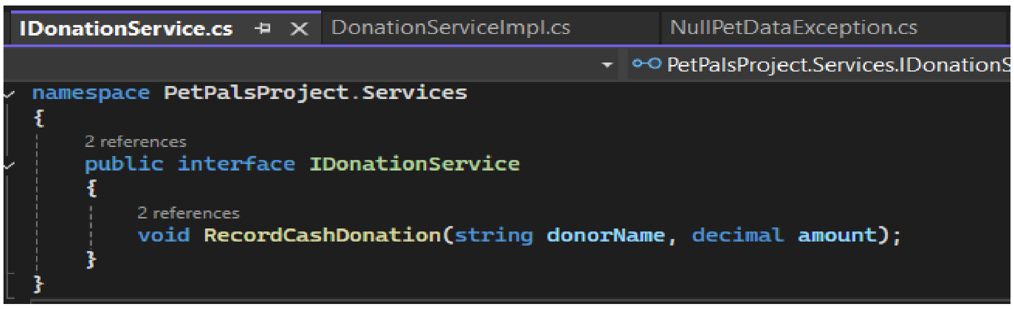
IPetService.cs – Interface for Pet Features



IPetService.cs   PetServiceImpl.cs   IAd

```
namespace PetPalsProject.Services
{
    using PetPalsProject.Models;

    2 references
    public interface IPetService
    {
        2 references
        void AddPet(Pet pet);
        2 references
        void ListAvailablePets();
    }
}
```

IDonationService.cs – Interface for Donations

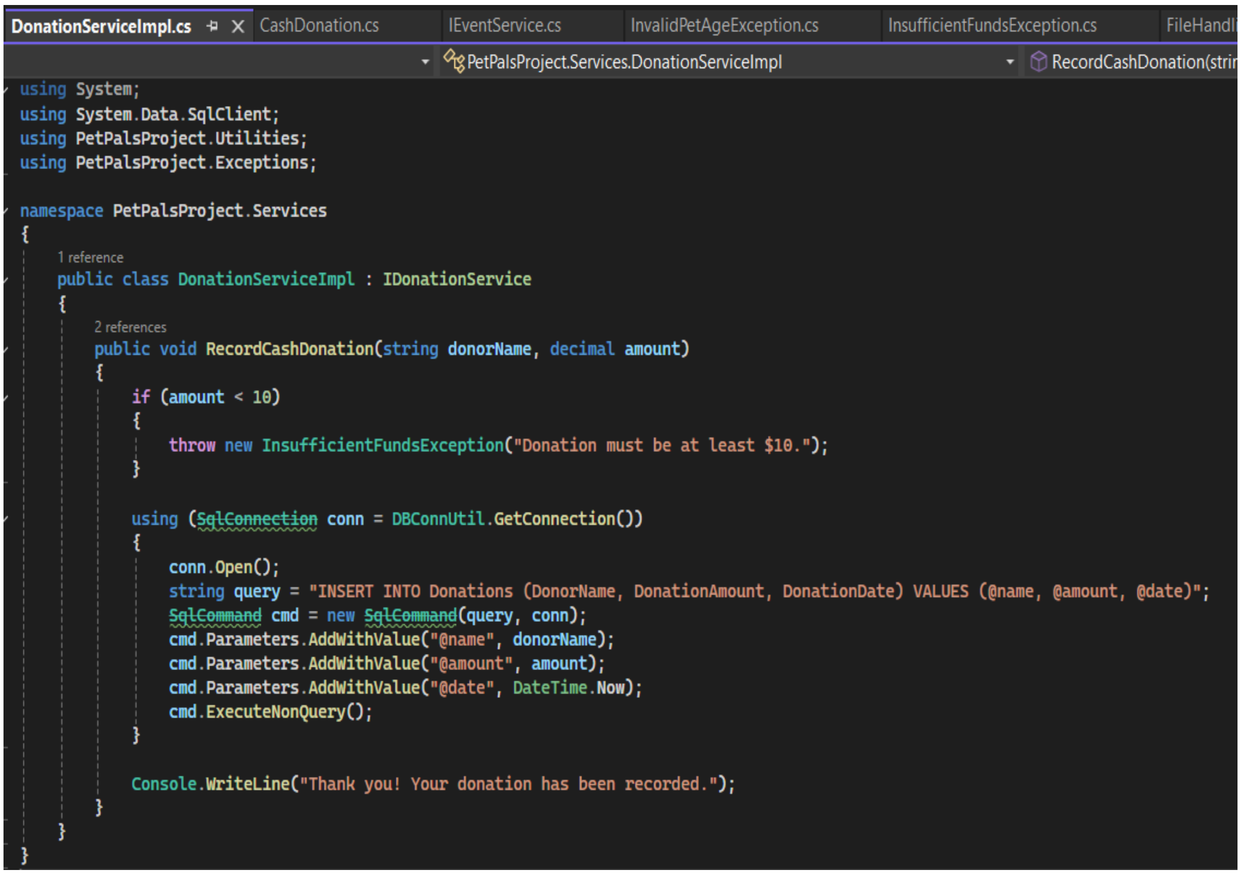


IDonationService.cs   DonationServiceImpl.cs   NullPetDataException.cs

PetPalsProject.Services.IDonationS

```
namespace PetPalsProject.Services
{
    2 references
    public interface IDonationService
    {
        2 references
        void RecordCashDonation(string donorName, decimal amount);
    }
}
```

DonationServiceImpl.cs

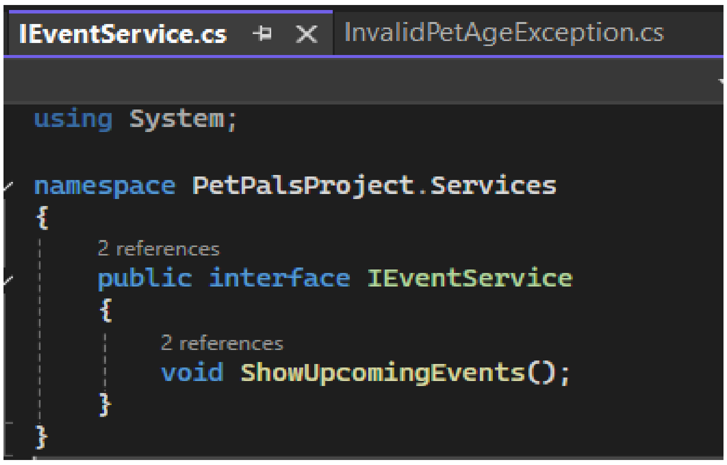


DonationServiceImpl.cs   CashDonation.cs   IEventService.cs   InvalidPetAgeException.cs   InsufficientFundsException.cs   FileHandl

PetPalsProject.Services.DonationServiceImpl   RecordCashDonation(strin

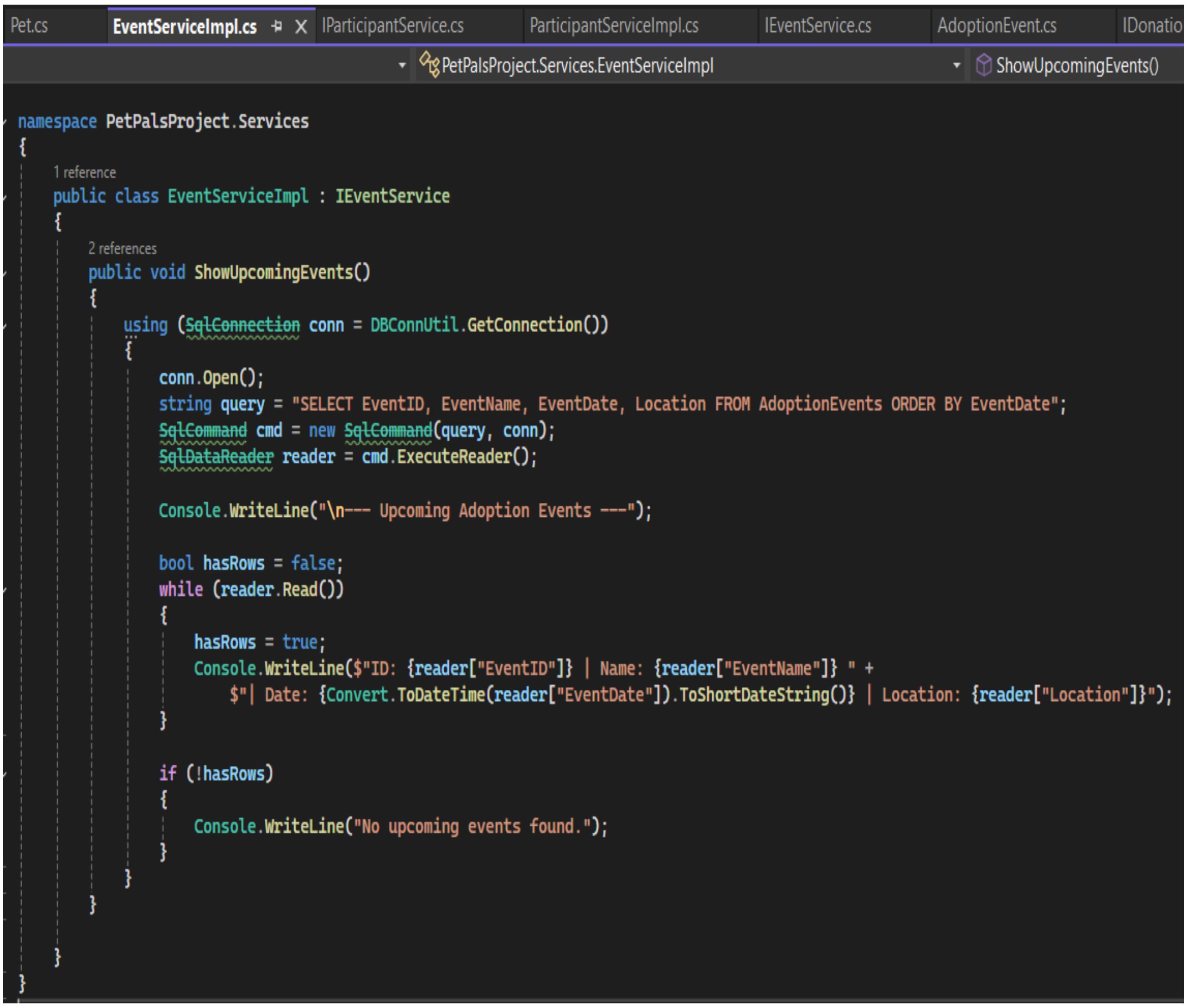
```
using System;
using System.Data.SqlClient;
using PetPalsProject.Utilities;
using PetPalsProject.Exceptions;

namespace PetPalsProject.Services
{
    1 reference
    public class DonationServiceImpl : IDonationService
    {
        2 references
        public void RecordCashDonation(string donorName, decimal amount)
        {
            if (amount < 10)
            {
                throw new InsufficientFundsException("Donation must be at least $10.");
            }

            using (SqlConnection conn = DBConnUtil.GetConnection())
            {
                conn.Open();
                string query = "INSERT INTO Donations (DonorName, DonationAmount, DonationDate) VALUES (@name, @amount, @date)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@name", donorName);
                cmd.Parameters.AddWithValue("@amount", amount);
                cmd.Parameters.AddWithValue("@date", DateTime.Now);
                cmd.ExecuteNonQuery();
            }

            Console.WriteLine("Thank you! Your donation has been recorded.");
        }
    }
}
```

IEventService.cs – Interface for Events



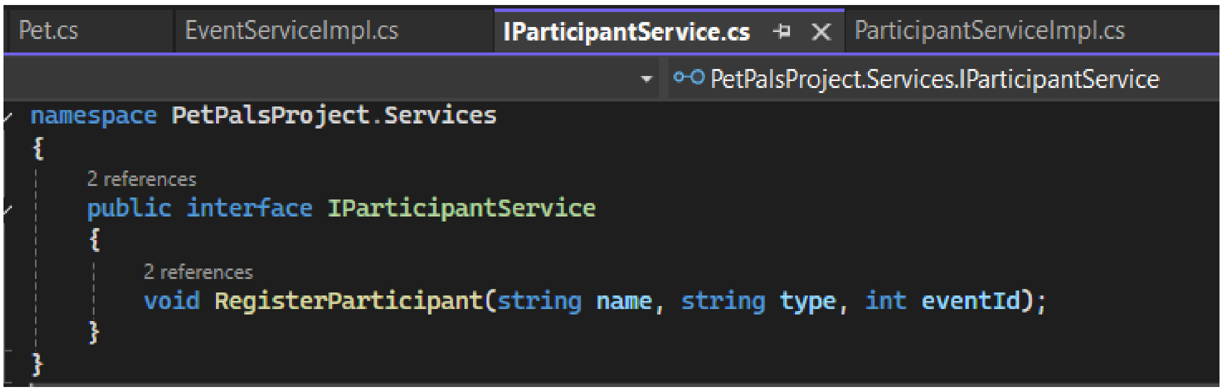
```
using System;

namespace PetPalsProject.Services
{
    2 references
    public interface IEventService
    {
        2 references
        void ShowUpcomingEvents();
    }
}
```

EventServiceImpl.cs – View Upcoming Events



```
namespace PetPalsProject.Services
{
    1 reference
    public class EventServiceImpl : IEventService
    {
        2 references
        public void ShowUpcomingEvents()
        {
            using (SqlConnection conn = DBConnUtil.GetConnection())
            {
                conn.Open();
                string query = "SELECT EventID, EventName, EventDate, Location FROM AdoptionEvents ORDER BY EventDate";
                SqlCommand cmd = new SqlCommand(query, conn);
                SqlDataReader reader = cmd.ExecuteReader();

                Console.WriteLine("\n--- Upcoming Adoption Events ---");

                bool hasRows = false;
                while (reader.Read())
                {
                    hasRows = true;
                    Console.WriteLine($"ID: {reader["EventID"]} | Name: {reader["EventName"]} " +
                        $"| Date: {Convert.ToDateTime(reader["EventDate"]).ToShortDateString()} | Location: {reader["Location"]}");
                }

                if (!hasRows)
                {
                    Console.WriteLine("No upcoming events found.");
                }
            }
        }

    }
}
```

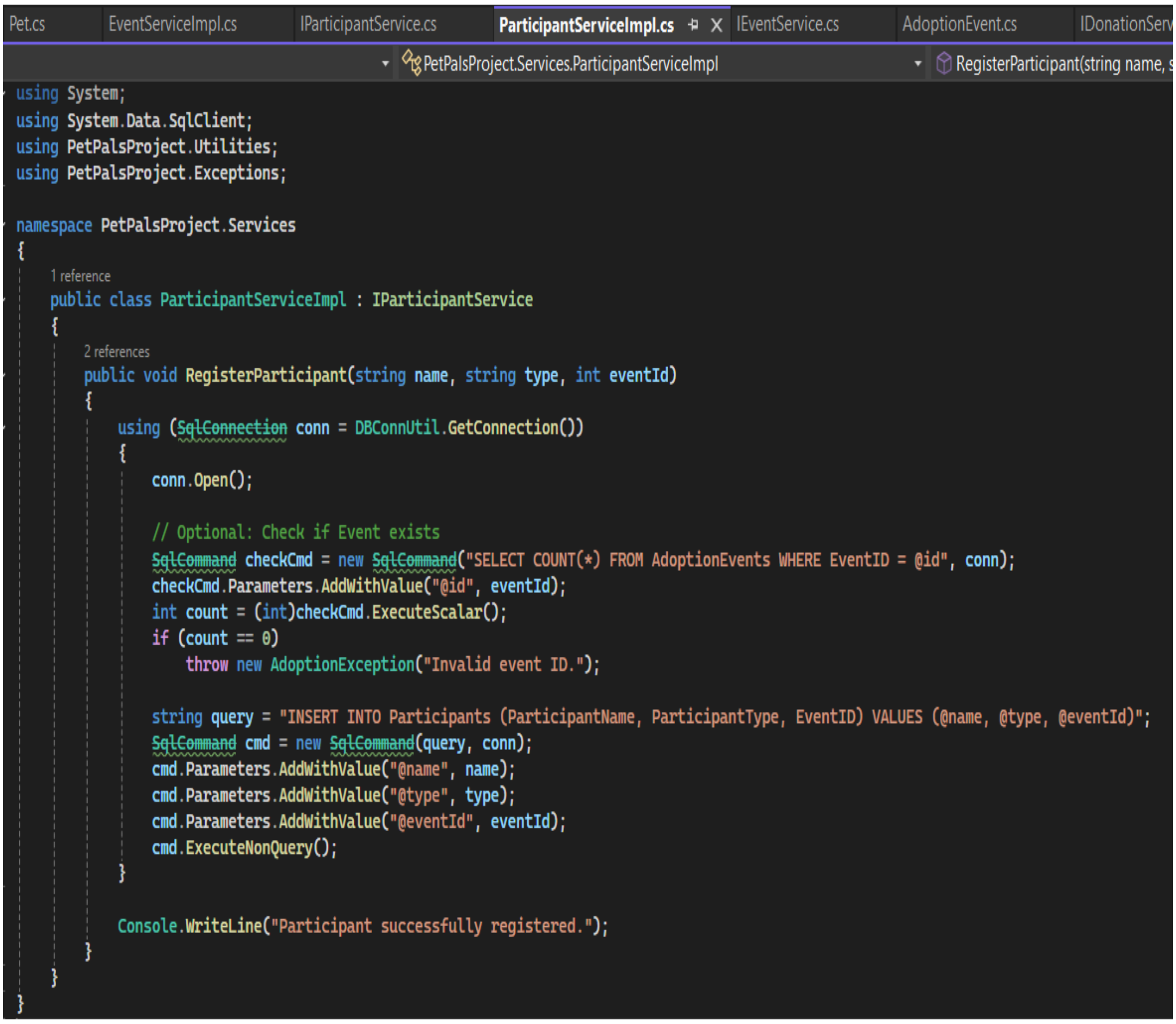
IParticipantService.cs – Interface



Pet.cs
EventServiceImpl.cs
IParticipantService.cs
ParticipantServiceImpl.cs
PetPalsProject.Services.IParticipantService

```
namespace PetPalsProject.Services
{
    2 references
    public interface IParticipantService
    {
        2 references
        void RegisterParticipant(string name, string type, int eventId);
    }
}
```

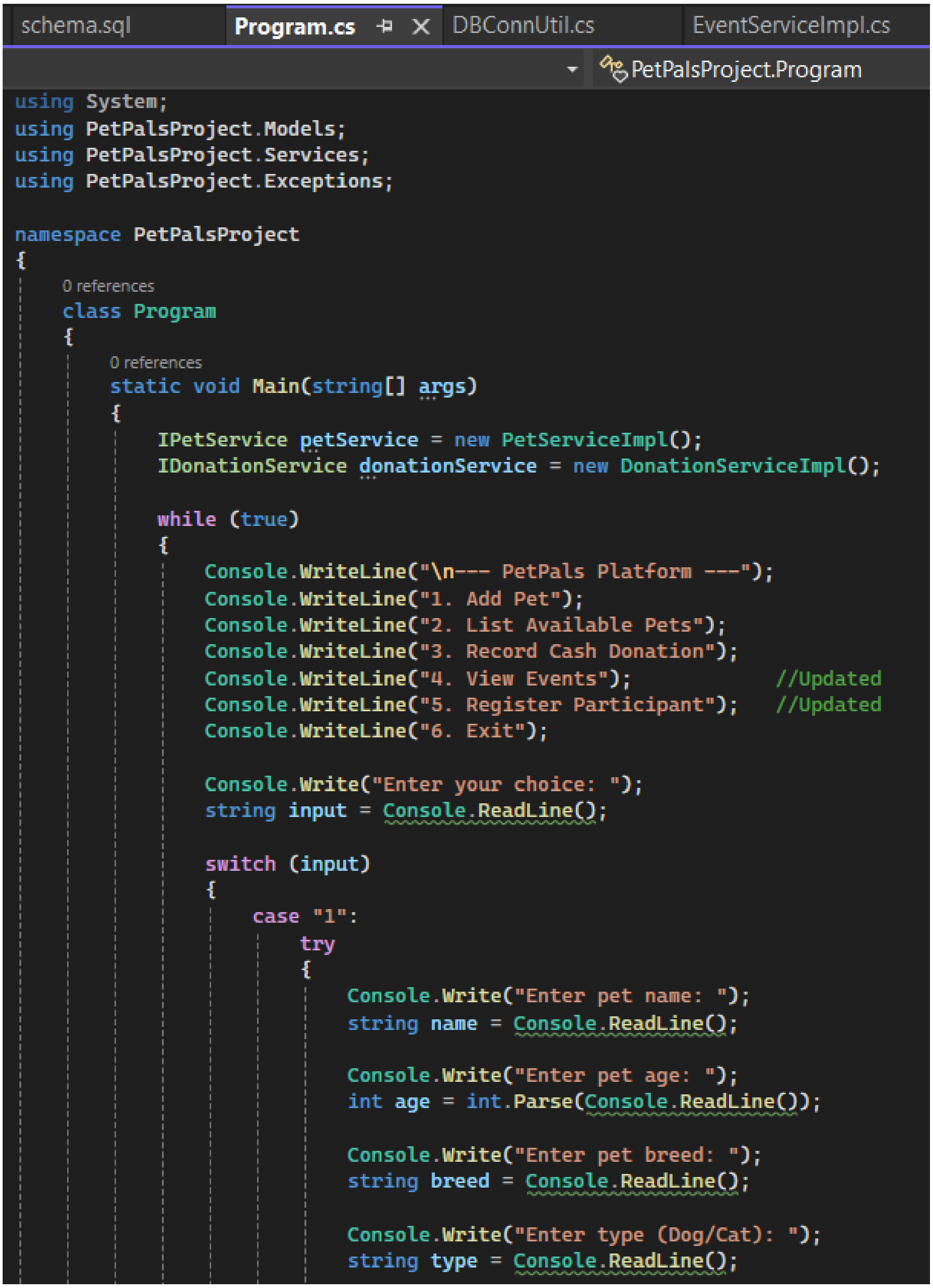
ParticipantServiceImpl.cs – Register Participant



Pet.cs
EventServiceImpl.cs
IParticipantService.cs
ParticipantServiceImpl.cs
IEventService.cs
AdoptionEvent.cs
IDonationServ
PetPalsProject.Services.ParticipantServiceImpl
RegisterParticipant(string name, s

```
using System;
using System.Data.SqlClient;
using PetPalsProject.Utilities;
using PetPalsProject.Exceptions;

namespace PetPalsProject.Services
{
    1 reference
    public class ParticipantServiceImpl : IParticipantService
    {
        2 references
        public void RegisterParticipant(string name, string type, int eventId)
        {
            using (SqlConnection conn = DBConnUtil.GetConnection())
            {
                conn.Open();

                // Optional: Check if Event exists
                SqlCommand checkCmd = new SqlCommand("SELECT COUNT(*) FROM AdoptionEvents WHERE EventID = @id", conn);
                checkCmd.Parameters.AddWithValue("@id", eventId);
                int count = (int)checkCmd.ExecuteScalar();
                if (count == 0)
                    throw new AdoptionException("Invalid event ID.");

                string query = "INSERT INTO Participants (ParticipantName, ParticipantType, EventID) VALUES (@name, @type, @eventId)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@type", type);
                cmd.Parameters.AddWithValue("@eventId", eventId);
                cmd.ExecuteNonQuery();
            }

            Console.WriteLine("Participant successfully registered.");
        }
    }
}
```

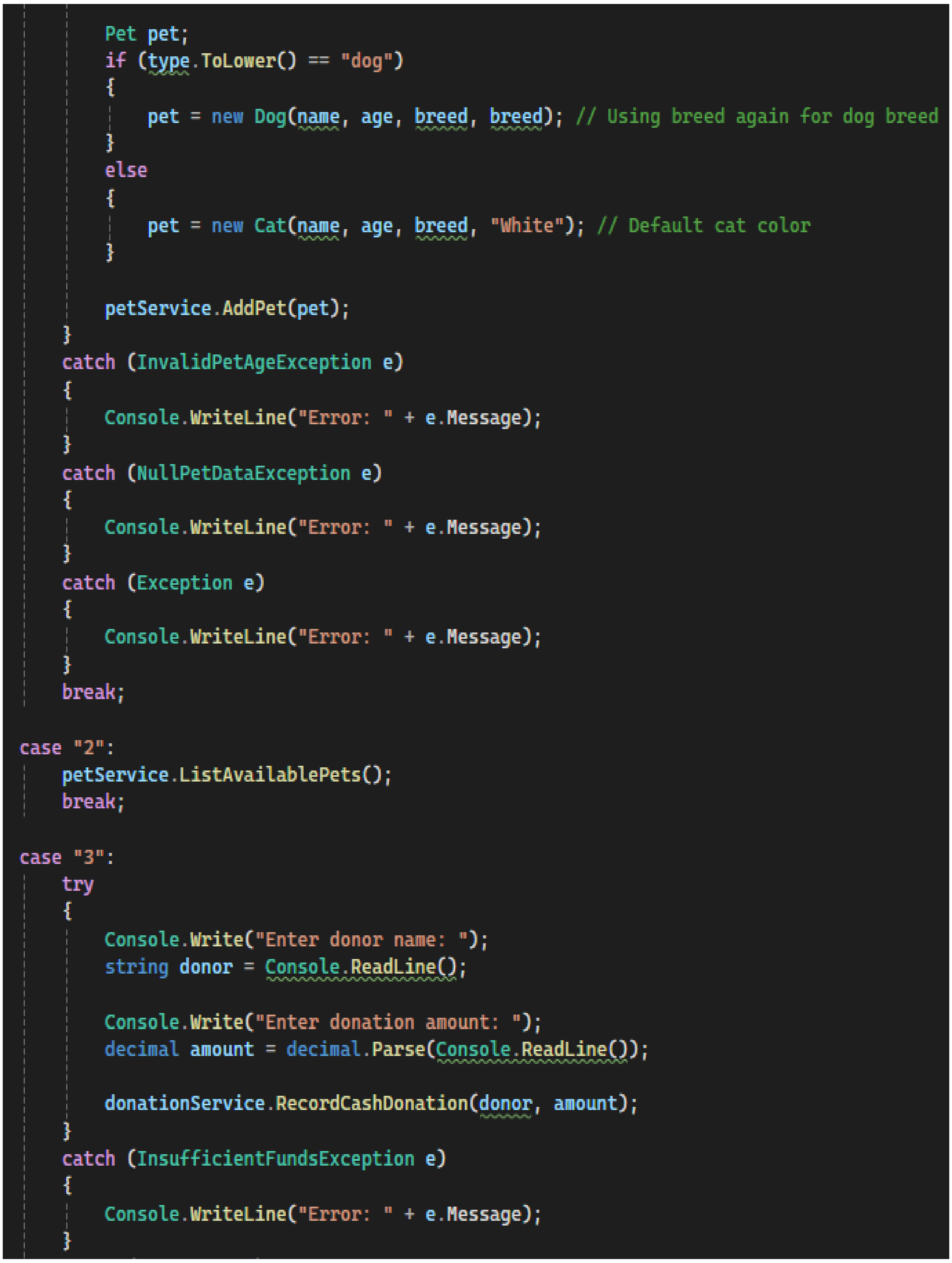
Program.cs – Console Menu-Driven UI



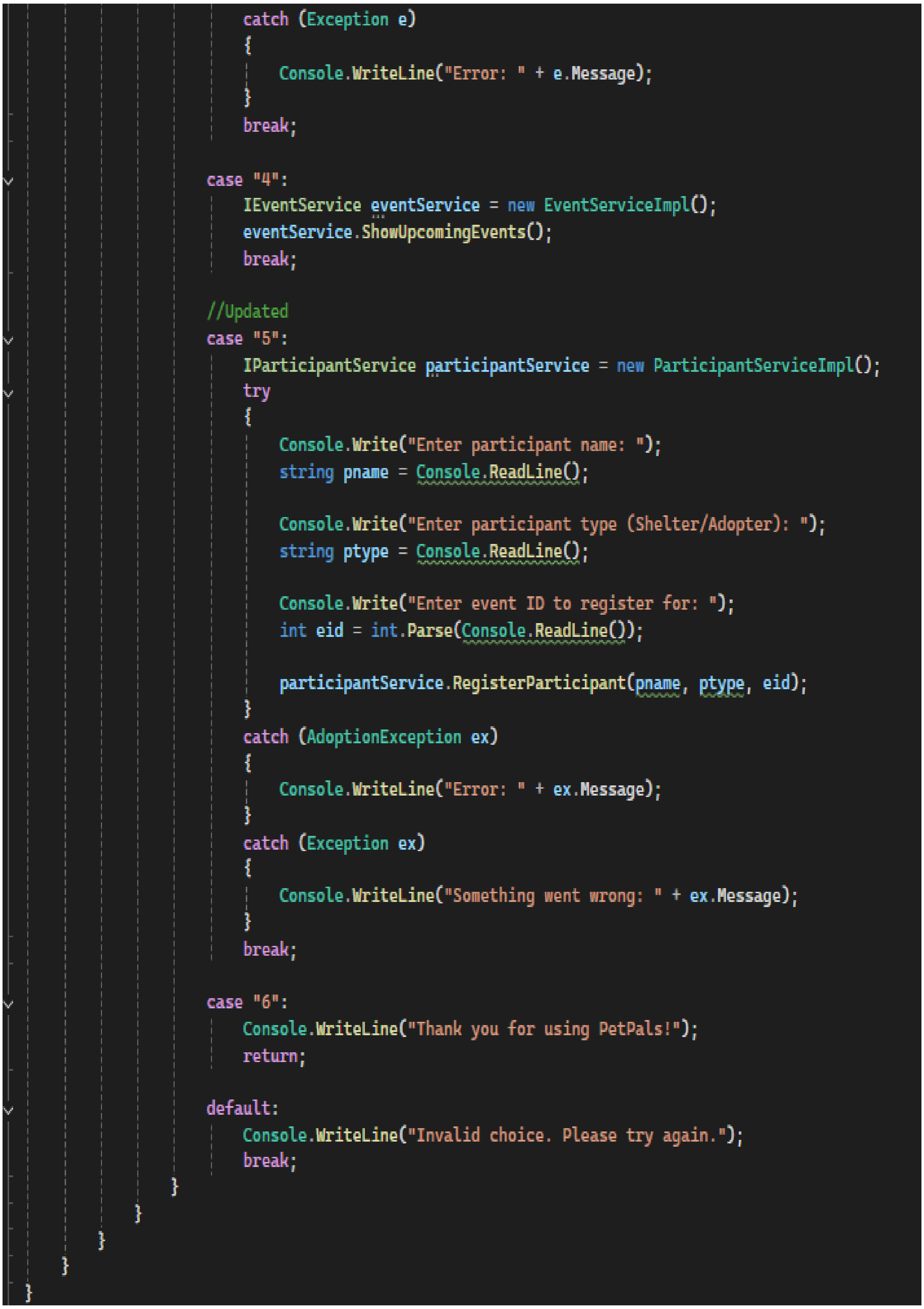
schema.sql | Program.cs | DBConnUtil.cs | EventServiceImpl.cs

PetPalsProject.Program

```
using System;
using PetPalsProject.Models;
using PetPalsProject.Services;
using PetPalsProject.Exceptions;

namespace PetPalsProject
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            IPetService petService = new PetServiceImpl();
            IDonationService donationService = new DonationServiceImpl();

            while (true)
            {
                Console.WriteLine("\n--- PetPals Platform ---");
                Console.WriteLine("1. Add Pet");
                Console.WriteLine("2. List Available Pets");
                Console.WriteLine("3. Record Cash Donation");
                Console.WriteLine("4. View Events");              //Updated
                Console.WriteLine("5. Register Participant");   //Updated
                Console.WriteLine("6. Exit");

                Console.Write("Enter your choice: ");
                string input = Console.ReadLine();

                switch (input)
                {
                    case "1":
                        try
                        {
                            Console.Write("Enter pet name: ");
                            string name = Console.ReadLine();

                            Console.Write("Enter pet age: ");
                            int age = int.Parse(Console.ReadLine());

                            Console.Write("Enter pet breed: ");
                            string breed = Console.ReadLine();

                            Console.Write("Enter type (Dog/Cat): ");
                            string type = Console.ReadLine();
```

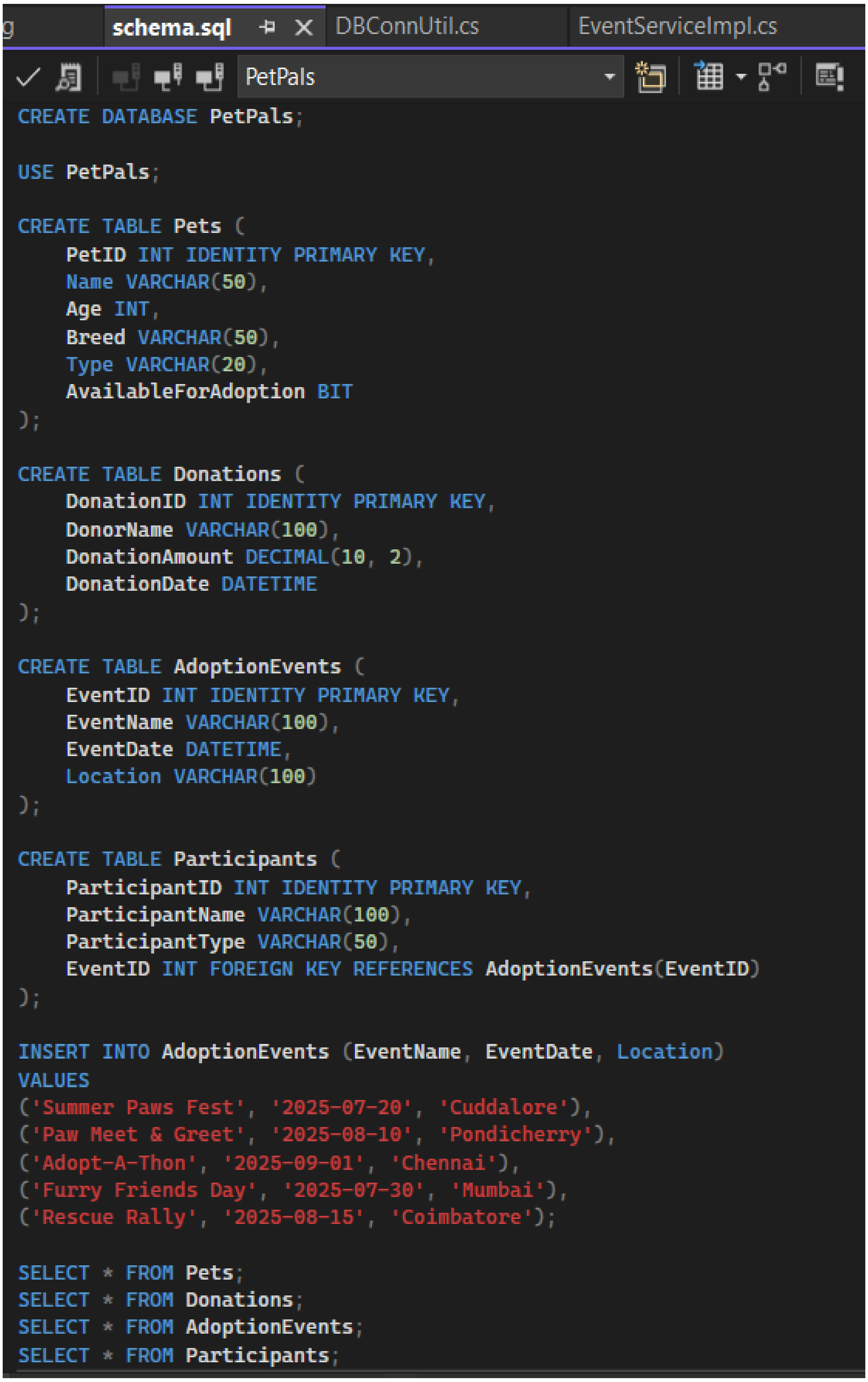


```
        Pet pet;
        if (type.ToLower() == "dog")
        {
            pet = new Dog(name, age, breed, breed); // Using breed again for dog breed
        }
        else
        {
            pet = new Cat(name, age, breed, "White"); // Default cat color
        }

        petService.AddPet(pet);
    }
    catch (InvalidPetAgeException e)
    {
        Console.WriteLine("Error: " + e.Message);
    }
    catch (NullPetDataException e)
    {
        Console.WriteLine("Error: " + e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: " + e.Message);
    }
    break;

case "2":
    petService.ListAvailablePets();
    break;

case "3":
    try
    {
        Console.Write("Enter donor name: ");
        string donor = Console.ReadLine();

        Console.Write("Enter donation amount: ");
        decimal amount = decimal.Parse(Console.ReadLine());

        donationService.RecordCashDonation(donor, amount);
    }
    catch (InsufficientFundsException e)
    {
        Console.WriteLine("Error: " + e.Message);
    }
```



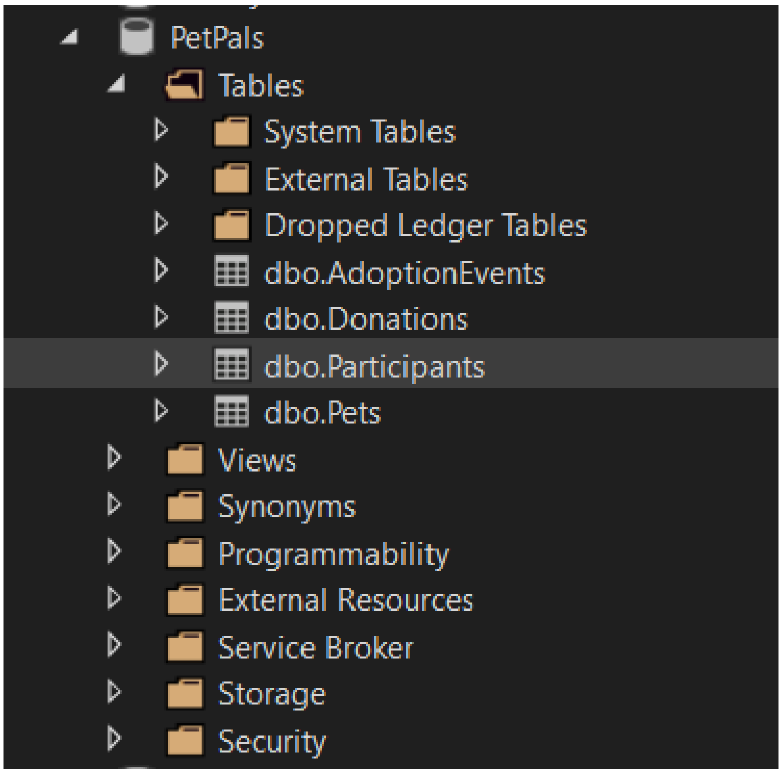
```
                        catch (Exception e)
                        {
                            Console.WriteLine("Error: " + e.Message);
                        }
                        break;

                    case "4":
                        IEventService eventService = new EventServiceImpl();
                        eventService.ShowUpcomingEvents();
                        break;

                    //Updated
                    case "5":
                        IParticipantService participantService = new ParticipantServiceImpl();
                        try
                        {
                            Console.Write("Enter participant name: ");
                            string pname = Console.ReadLine();

                            Console.Write("Enter participant type (Shelter/Adopter): ");
                            string ptype = Console.ReadLine();

                            Console.Write("Enter event ID to register for: ");
                            int eid = int.Parse(Console.ReadLine());

                            participantService.RegisterParticipant(pname, ptype, eid);
                        }
                        catch (AdoptionException ex)
                        {
                            Console.WriteLine("Error: " + ex.Message);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine("Something went wrong: " + ex.Message);
                        }
                        break;

                    case "6":
                        Console.WriteLine("Thank you for using PetPals!");
                        return;

                    default:
                        Console.WriteLine("Invalid choice. Please try again.");
                        break;
                }
            }
        }
    }
}
```

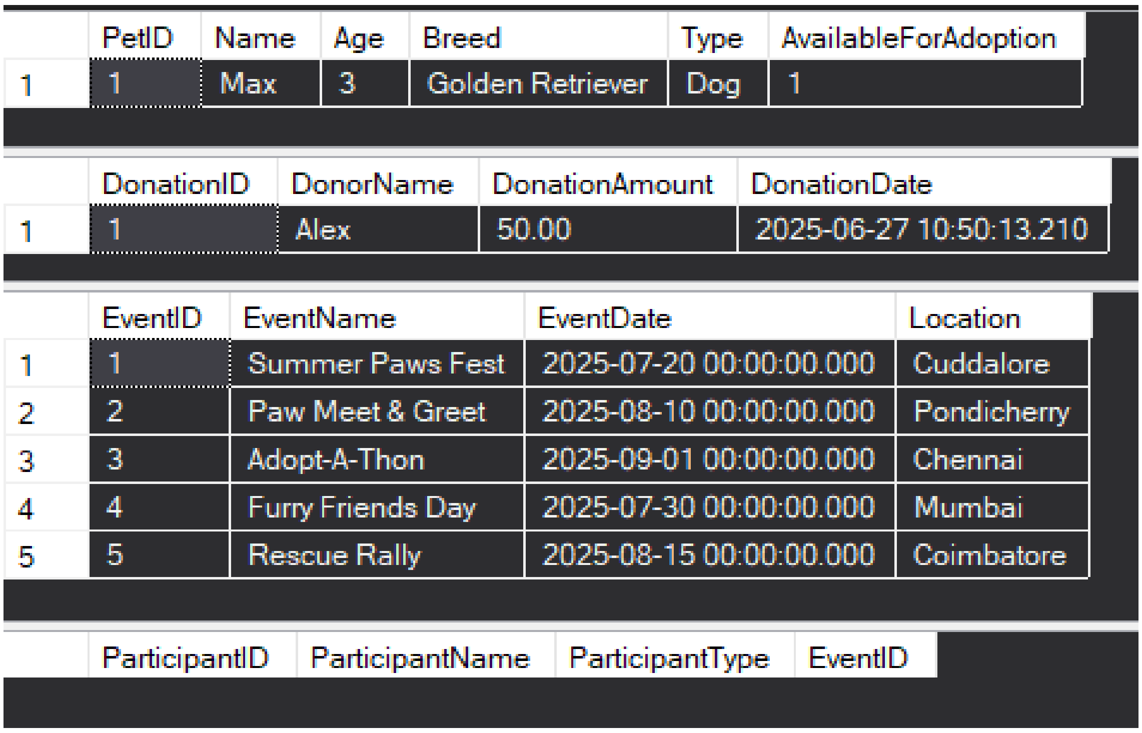
SQL Table Creation (Schema)



```sql
CREATE DATABASE PetPals;

USE PetPals;

CREATE TABLE Pets (
    PetID INT IDENTITY PRIMARY KEY,
    Name VARCHAR(50),
    Age INT,
    Breed VARCHAR(50),
    Type VARCHAR(20),
    AvailableForAdoption BIT
);

CREATE TABLE Donations (
    DonationID INT IDENTITY PRIMARY KEY,
    DonorName VARCHAR(100),
    DonationAmount DECIMAL(10, 2),
    DonationDate DATETIME
);

CREATE TABLE AdoptionEvents (
    EventID INT IDENTITY PRIMARY KEY,
    EventName VARCHAR(100),
    EventDate DATETIME,
    Location VARCHAR(100)
);

CREATE TABLE Participants (
    ParticipantID INT IDENTITY PRIMARY KEY,
    ParticipantName VARCHAR(100),
    ParticipantType VARCHAR(50),
    EventID INT FOREIGN KEY REFERENCES AdoptionEvents(EventID)
);

INSERT INTO AdoptionEvents (EventName, EventDate, Location)
VALUES
('Summer Paws Fest', '2025-07-20', 'Cuddalore'),
('Paw Meet & Greet', '2025-08-10', 'Pondicherry'),
('Adopt-A-Thon', '2025-09-01', 'Chennai'),
('Furry Friends Day', '2025-07-30', 'Mumbai'),
('Rescue Rally', '2025-08-15', 'Coimbatore');

SELECT * FROM Pets;
SELECT * FROM Donations;
SELECT * FROM AdoptionEvents;
SELECT * FROM Participants;
```

SQL Server Object Explorer – PetPals DB View





	PetID	Name	Age	Breed	Type	AvailableForAdoption
1	1	Max	3	Golden Retriever	Dog	1

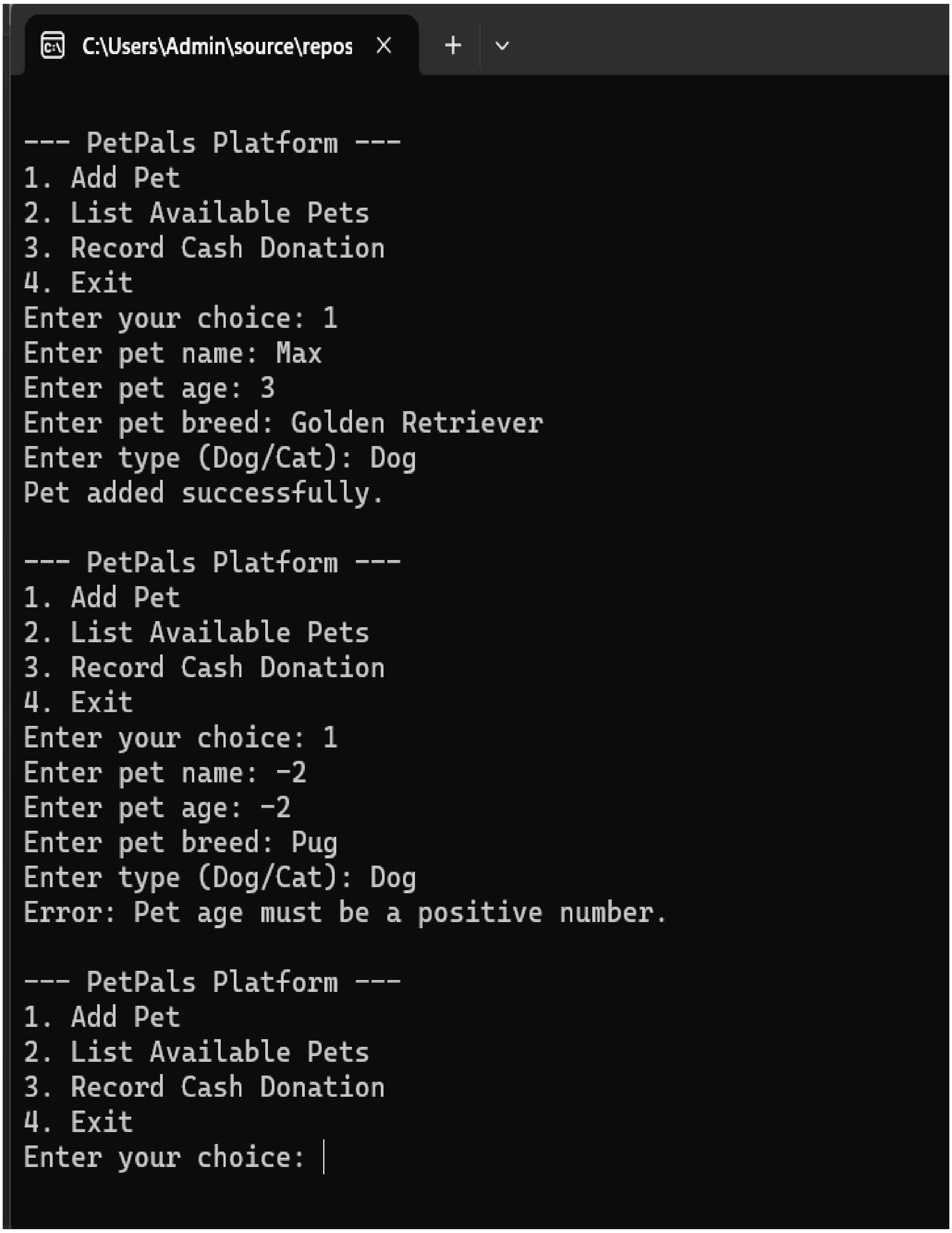
	DonationID	DonorName	DonationAmount	DonationDate
1	1	Alex	50.00	2025-06-27 10:50:13.210

	EventID	EventName	EventDate	Location
1	1	Summer Paws Fest	2025-07-20 00:00:00.000	Cuddalore
2	2	Paw Meet & Greet	2025-08-10 00:00:00.000	Pondicherry
3	3	Adopt-A-Thon	2025-09-01 00:00:00.000	Chennai
4	4	Furry Friends Day	2025-07-30 00:00:00.000	Mumbai
5	5	Rescue Rally	2025-08-15 00:00:00.000	Coimbatore

	ParticipantID	ParticipantName	ParticipantType	EventID

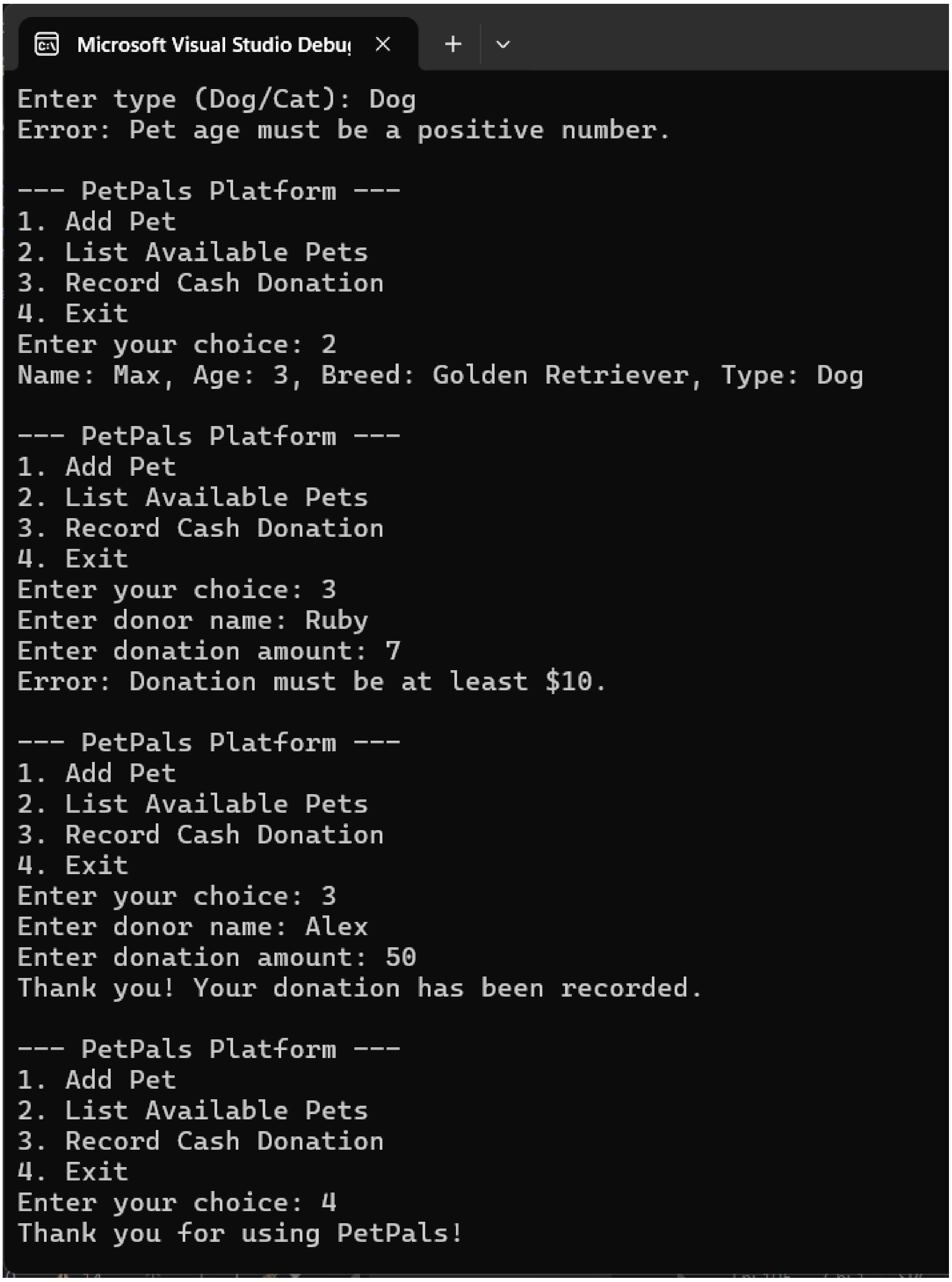
OUTPUT:

Sample Output – Add Pet, List Pets



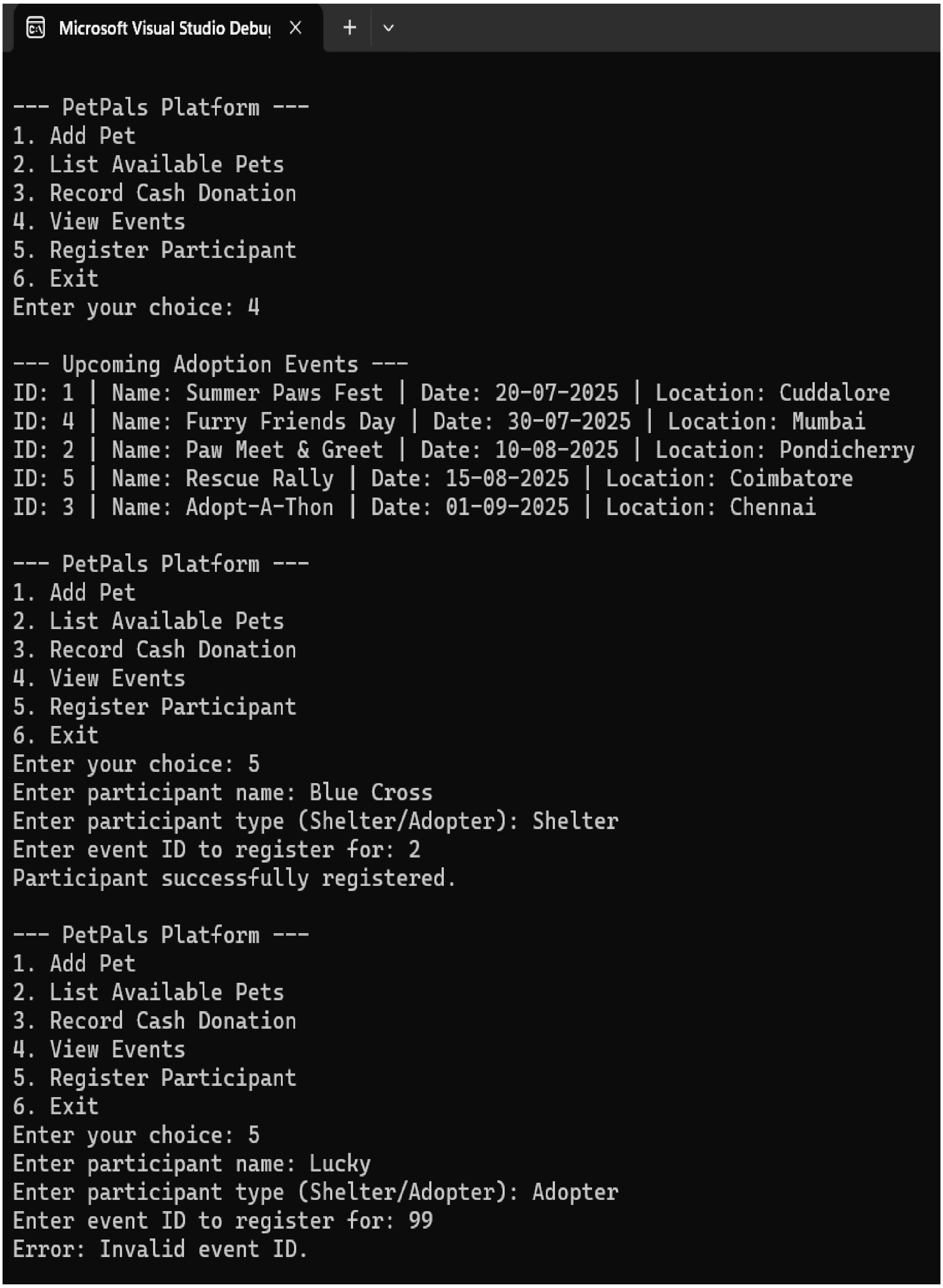
```
--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 1
Enter pet name: Max
Enter pet age: 3
Enter pet breed: Golden Retriever
Enter type (Dog/Cat): Dog
Pet added successfully.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 1
Enter pet name: -2
Enter pet age: -2
Enter pet breed: Pug
Enter type (Dog/Cat): Dog
Error: Pet age must be a positive number.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice:
```

Sample Output – Record Donation

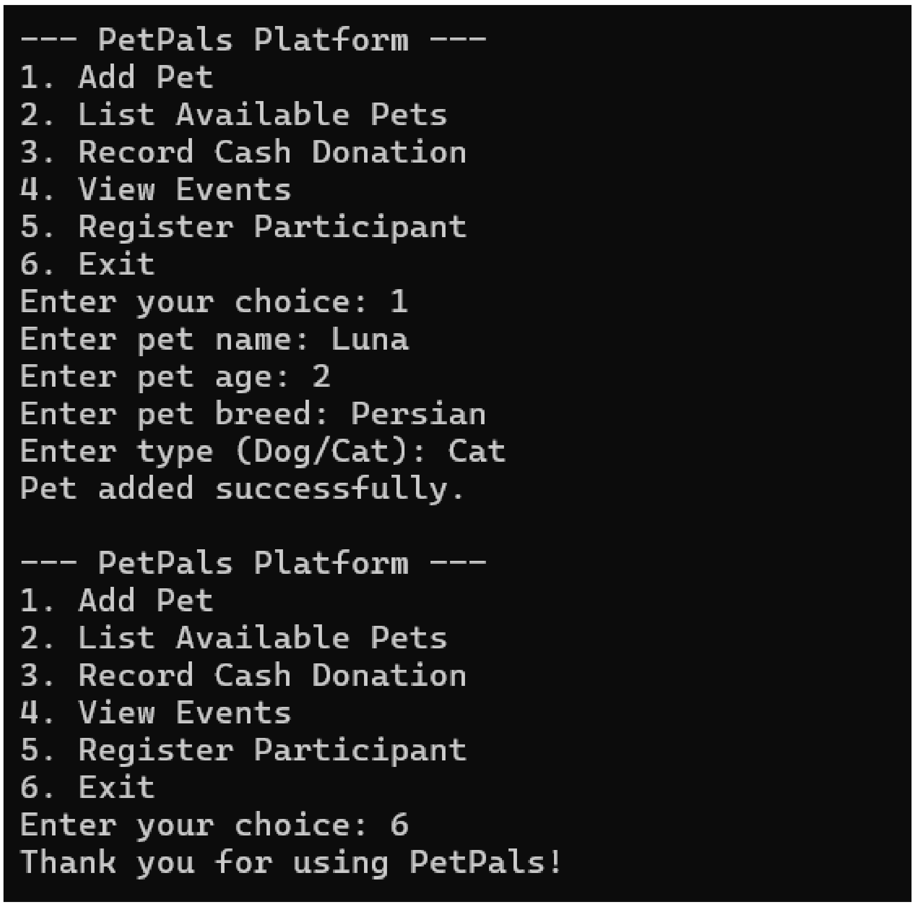


```
Enter type (Dog/Cat): Dog
Error: Pet age must be a positive number.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 2
Name: Max, Age: 3, Breed: Golden Retriever, Type: Dog

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 3
Enter donor name: Ruby
Enter donation amount: 7
Error: Donation must be at least $10.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 3
Enter donor name: Alex
Enter donation amount: 50
Thank you! Your donation has been recorded.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. Exit
Enter your choice: 4
Thank you for using PetPals!
```

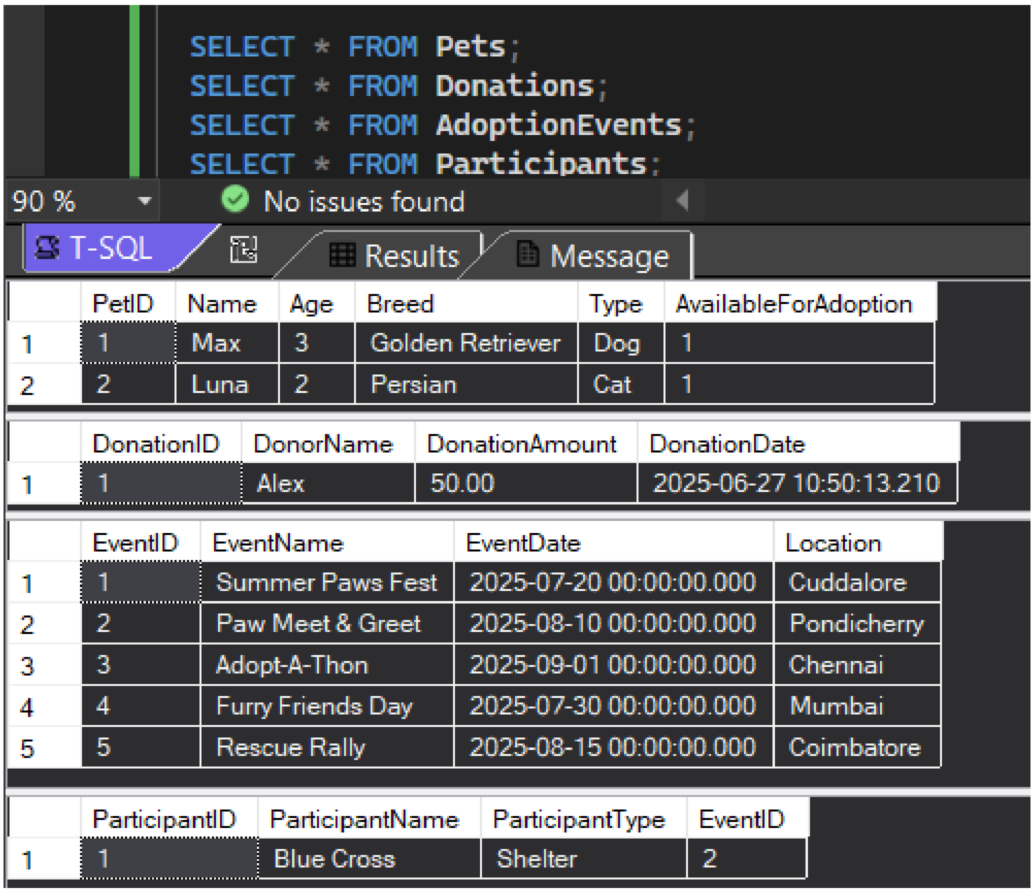
Sample Output – View Events and Register Participant



```
--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. View Events
5. Register Participant
6. Exit
Enter your choice: 4

--- Upcoming Adoption Events ---
ID: 1 | Name: Summer Paws Fest | Date: 20-07-2025 | Location: Cuddalore
ID: 4 | Name: Furry Friends Day | Date: 30-07-2025 | Location: Mumbai
ID: 2 | Name: Paw Meet & Greet | Date: 10-08-2025 | Location: Pondicherry
ID: 5 | Name: Rescue Rally | Date: 15-08-2025 | Location: Coimbatore
ID: 3 | Name: Adopt-A-Thon | Date: 01-09-2025 | Location: Chennai

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. View Events
5. Register Participant
6. Exit
Enter your choice: 5
Enter participant name: Blue Cross
Enter participant type (Shelter/Adopter): Shelter
Enter event ID to register for: 2
Participant successfully registered.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. View Events
5. Register Participant
6. Exit
Enter your choice: 5
Enter participant name: Lucky
Enter participant type (Shelter/Adopter): Adopter
Enter event ID to register for: 99
Error: Invalid event ID.
```



```
--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. View Events
5. Register Participant
6. Exit
Enter your choice: 1
Enter pet name: Luna
Enter pet age: 2
Enter pet breed: Persian
Enter type (Dog/Cat): Cat
Pet added successfully.

--- PetPals Platform ---
1. Add Pet
2. List Available Pets
3. Record Cash Donation
4. View Events
5. Register Participant
6. Exit
Enter your choice: 6
Thank you for using PetPals!
```



```sql
SELECT * FROM Pets;
SELECT * FROM Donations;
SELECT * FROM AdoptionEvents;
SELECT * FROM Participants;
```

	PetID	Name	Age	Breed	Type	AvailableForAdoption
1	1	Max	3	Golden Retriever	Dog	1
2	2	Luna	2	Persian	Cat	1

	DonationID	DonorName	DonationAmount	DonationDate
1	1	Alex	50.00	2025-06-27 10:50:13.210

	EventID	EventName	EventDate	Location
1	1	Summer Paws Fest	2025-07-20 00:00:00.000	Cuddalore
2	2	Paw Meet & Greet	2025-08-10 00:00:00.000	Pondicherry
3	3	Adopt-A-Thon	2025-09-01 00:00:00.000	Chennai
4	4	Furry Friends Day	2025-07-30 00:00:00.000	Mumbai
5	5	Rescue Rally	2025-08-15 00:00:00.000	Coimbatore

	ParticipantID	ParticipantName	ParticipantType	EventID
1	1	Blue Cross	Shelter	2