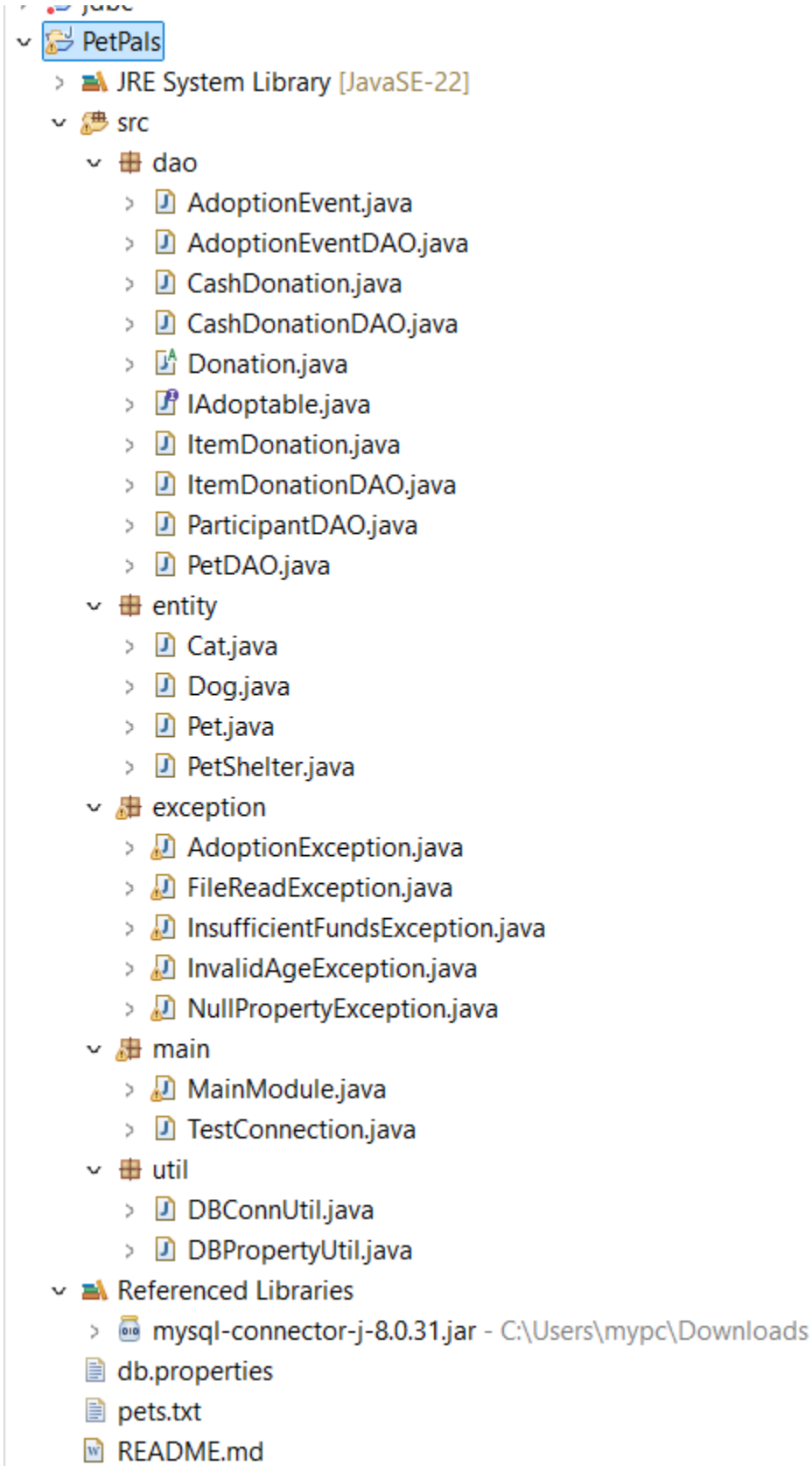
Project Structure



dao package :

IAdoptable.java (Interface):

```
package dao;
public interface IAdoptable {
  void adopt() throws Exception;
}
```

==========

AdoptionEvent.java:

```
package dao;
import java.util.ArrayList;
import java.util.List;
import entity.PetShelter;
public class AdoptionEvent {
  private List<IAdoptable> participants;
  public AdoptionEvent() {
    participants = new ArrayList<>();
  }
  public void registerParticipant(PetShelter shelter) {
    participants.add((IAdoptable) shelter);
    System.out.println("Participant registered for the adoption event.");
  }
  public void hostEvent() {
    System.out.println("Hosting Adoption Event...");
    for (IAdoptable participant : participants) {
      try {
        participant.adopt();
      } catch (Exception e) {
        System.out.println("Error during adoption: " + e.getMessage());
      }
    }
    System.out.println("Adoption Event concluded.");
  }
  public List<IAdoptable> getParticipants() {
    return participants;
```

```
  }
}
```

==========

AdoptionEventDAO.java:

```
package dao;
import util.DBConnUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
public class AdoptionEventDAO {

  public int hostEvent(String description) {
      String query = "INSERT INTO adoptionevents (description, event_date) VALUES
(?, CURDATE())";
      int generatedId = -1;
      try (Connection conn = DBConnUtil.getConnection("db.properties");
          PreparedStatement stmt = conn.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS)) {
          stmt.setString(1, description);
          stmt.executeUpdate();
          ResultSet rs = stmt.getGeneratedKeys();
          if (rs.next()) {
              generatedId = rs.getInt(1);
              System.out.println(" Adoption event recorded in DB with ID: " + generatedId);
          }
      } catch (SQLException e) {
          System.out.println(" Error recording event: " + e.getMessage());
      }
      return generatedId;
  }

  public int hostEvent(String description, String location) {
      String query = "INSERT INTO adoptionevents (description, location, event_date)
VALUES (?, ?, CURDATE())";
      int generatedId = -1;
```

```
    try (Connection conn = DBConnUtil.getConnection("db.properties");
         PreparedStatement stmt = conn.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS)) {
        stmt.setString(1, description);
        stmt.setString(2, location);
        stmt.executeUpdate();
        ResultSet rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            generatedId = rs.getInt(1);
            System.out.println("Adoption event recorded in DB with ID: " + generatedId);
        }
    } catch (SQLException e) {
        System.out.println("Error recording event: " + e.getMessage());
    }
    return generatedId;
  }
}
```

==========

Donatio.java

```
package dao;
import exception.InsufficientFundsException;
public abstract class Donation {
  protected String donorName;
  protected double amount;
  // Constructor
  public Donation(String donorName, double amount) {
    this.donorName = donorName;
    this.amount = amount;
  }
  // Getters and Setters
  public String getDonorName() {
    return donorName;
  }
  public void setDonorName(String donorName) {
    this.donorName = donorName;
  }
```

```
  public double getAmount() {
      return amount;
  }
  public void setAmount(double amount) {
      this.amount = amount;
  }
  // Abstract Method with exception
  public abstract void recordDonation() throws InsufficientFundsException;
}
```

==========

CashDonation.java

```
package dao;
import exception.InsufficientFundsException;
import util.DBConnUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.Date;
public class CashDonation extends Donation {
  private LocalDateTime donationDate;
  public CashDonation(String donorName, double amount, Date date) {
      super(donorName, amount);
      this.donationDate = LocalDateTime.ofInstant(date.toInstant(),
ZoneId.systemDefault());
  }
  public Date getDonationDate() {
      return Date.from(donationDate.atZone(ZoneId.systemDefault()).toInstant());
  }
  public void setDonationDate(LocalDateTime donationDate) {
      this.donationDate = donationDate;
  }
  @Override
  public void recordDonation() throws InsufficientFundsException {
```

```
        if (amount < 10) {
            throw new InsufficientFundsException("Donation amount must be at least
₹10.");
        }
        System.out.println("Cash Donation Recorded:");
        System.out.println("Donor: " + donorName);
        System.out.println("Amount: ₹" + amount);
        System.out.println("Date: " + donationDate);
        Connection conn = DBConnUtil.getConnection("db.properties");
        if (conn == null) {
            System.out.println("Database connection failed.");
            return;
        }
        String query = "INSERT INTO cashdonations (donor_name, amount,
donation_date) VALUES (?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, donorName);
            stmt.setDouble(2, amount);
            stmt.setDate(3, java.sql.Date.valueOf(donationDate.toLocalDate()));
            stmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Failed to insert donation: " + e.getMessage());
        }
    }
}
```

==========

CashDonationDAO.java

```
package dao;
import util.DBConnUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Date;
public class CashDonationDAO {
    public void addDonation(String donorName, double amount) {
        Connection conn = DBConnUtil.getConnection("db.properties");
        if (conn == null) {
            System.out.println("Database connection failed.");
            return;
```

```
        }
        String query = "INSERT INTO cashdonations (donor_name, amount,
donation_date) VALUES (?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, donorName);
            stmt.setDouble(2, amount);
            stmt.setDate(3, new Date(System.currentTimeMillis())); // current date
            stmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Failed to insert cash donation: " + e.getMessage());
        }
    }
}
```

==========
======

ItemDonation.java:

```
package dao;
public class ItemDonation extends Donation {
    private String itemType;
    // Constructor
    public ItemDonation(String donorName, double amount, String itemType) {
        super(donorName, amount);
        this.itemType = itemType;
    }
    // Getter and Setter
    public String getItemType() {
        return itemType;
    }
    public void setItemType(String itemType) {
        this.itemType = itemType;
    }
    // Implementation of abstract method
    @Override
    public void recordDonation() {
        System.out.println("Item Donation Recorded:");
```

```
        System.out.println("Donor: " + donorName);
        System.out.println("Amount (Estimated): $" + amount);
        System.out.println("Item Type: " + itemType);
    }
}
```

==========

```
package dao;
import util.DBConnUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class ItemDonationDAO {
    public void addDonation(String donorName, String itemDescription) {
        String query = "INSERT INTO itemdonations (donor_name, item_description,
donation_date) VALUES (?, ?, CURDATE())";
        try (Connection conn = DBConnUtil.getConnection("db.properties");
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, donorName);
            stmt.setString(2, itemDescription);
            stmt.executeUpdate();
            System.out.println(" Item donation inserted in database.");
        } catch (SQLException e) {
            System.out.println(" Error adding item donation: " + e.getMessage());
        }
    }
}
```

==========
======

ParticipantDAO.java:

```
package dao;
import util.DBConnUtil;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
```

```java
public class ParticipantDAO {

  public void addParticipant(int eventId, String shelterName) {
    Connection conn = DBConnUtil.getConnection("db.properties");
    if (conn == null) {
      System.out.println("Database connection failed.");
      return;
    }
    String query = "INSERT INTO eventparticipants (event_id, shelter_name) VALUES
(?, ?)";
    try (PreparedStatement stmt = conn.prepareStatement(query)) {
      stmt.setInt(1, eventId);
      stmt.setString(2, shelterName);
      stmt.executeUpdate();
    } catch (SQLException e) {
      System.out.println("Error adding participant: " + e.getMessage());
    }
  }

  public List<String> getParticipants(int eventId) {
    List<String> participants = new ArrayList<>();
    Connection conn = DBConnUtil.getConnection("db.properties");
    if (conn == null) {
      System.out.println("Database connection failed.");
      return participants;
    }
    String query = "SELECT shelter_name FROM eventparticipants WHERE event_id
= ?";
    try (PreparedStatement stmt = conn.prepareStatement(query)) {
      stmt.setInt(1, eventId);
      ResultSet rs = stmt.executeQuery();
      while (rs.next()) {
        participants.add(rs.getString("shelter_name"));
      }
    } catch (SQLException e) {
      System.out.println("Error retrieving participants: " + e.getMessage());
    }
    return participants;
  }
}
```

==========

PetDAO.java:

```
package dao;
import entity.Cat;
import entity.Dog;
import entity.Pet;
import util.DBConnUtil;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class PetDAO {

  public void addPet(Pet pet) {
      String query = "INSERT INTO pets (name, age, breed, dog_breed, cat_color, type)
VALUES (?, ?, ?, ?, ?, ?)";
      try (Connection conn = DBConnUtil.getConnection("db.properties");
           PreparedStatement stmt = conn.prepareStatement(query)) {
          stmt.setString(1, pet.getName());
          stmt.setInt(2, pet.getAge());
          stmt.setString(3, pet.getBreed());
          if (pet instanceof Dog) {
              stmt.setString(4, ((Dog) pet).getDogBreed());
              stmt.setNull(5, Types.VARCHAR);
              stmt.setString(6, "dog");
          } else if (pet instanceof Cat) {
              stmt.setNull(4, Types.VARCHAR);
              stmt.setString(5, ((Cat) pet).getCatColor());
              stmt.setString(6, "cat");
          } else {
              stmt.setNull(4, Types.VARCHAR);
              stmt.setNull(5, Types.VARCHAR);
              stmt.setString(6, "unknown");
          }
          stmt.executeUpdate();
          System.out.println(" Pet added to the database.");
      } catch (SQLException e) {
          System.out.println(" Error adding pet: " + e.getMessage());
      }
```

```
}

public List<Pet> getAllPets() {
    List<Pet> pets = new ArrayList<>();
    String query = "SELECT * FROM pets";
    try (Connection conn = DBConnUtil.getConnection("db.properties");
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(query)) {
        while (rs.next()) {
            String name = rs.getString("name");
            int age = rs.getInt("age");
            String breed = rs.getString("breed");
            String type = rs.getString("type");
            Pet pet;
            if ("dog".equalsIgnoreCase(type)) {
                pet = new Dog(name, age, breed, rs.getString("dog_breed"));
            } else if ("cat".equalsIgnoreCase(type)) {
                pet = new Cat(name, age, breed, rs.getString("cat_color"));
            } else {
                continue;
            }
            pets.add(pet);
        }
    } catch (SQLException e) {
        System.out.println(" Error reading pets: " + e.getMessage());
    }
    return pets;
}

public void removePetByName(String name) {
    String query = "DELETE FROM pets WHERE name = ?";
    try (Connection conn = DBConnUtil.getConnection("db.properties");
         PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, name);
        int count = stmt.executeUpdate();
        if (count > 0) {
            System.out.println(" Pet removed from the database.");
        } else {
            System.out.println(" Pet not found in database.");
        }
```

```
        } catch (SQLException e) {
            System.out.println(" Error removing pet: " + e.getMessage());
        }
    }
}
```

==========
======

Entity package:

pets.java:

```
package entity;
public class Pet {
    private String name;
    private int age;
    private String breed;
    // Constructor
    public Pet(String name, int age, String breed) {
        this.name = name;
        this.age = age;
        this.breed = breed;
    }
    // Getters
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public String getBreed() {
        return breed;
    }
    // Setters
    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
```

```
        this.age = age;
    }
    public void setBreed(String breed) {
        this.breed = breed;
    }
    // toString Method
    @Override
    public String toString() {
        return "Pet [Name=" + name + ", Age=" + age + ", Breed=" + breed + "]";
    }
}
```

==========

Dog.java:

```
package entity;
public class Dog extends Pet {
    private String dogBreed;
    // Constructor
    public Dog(String name, int age, String breed, String dogBreed) {
        super(name, age, breed);
        this.dogBreed = dogBreed;
    }
    // Getter
    public String getDogBreed() {
        return dogBreed;
    }
    // Setter
    public void setDogBreed(String dogBreed) {
        this.dogBreed = dogBreed;
    }
    // toString Method
    @Override
    public String toString() {
        return super.toString() + ", DogBreed=" + dogBreed;
    }
```

```
}
```

==========

Cat.java

```
package entity;
public class Cat extends Pet {
  private String catColor;
  // Constructor
  public Cat(String name, int age, String breed, String catColor) {
     super(name, age, breed);
     this.catColor = catColor;
  }
  // Getter
  public String getCatColor() {
     return catColor;
  }
  // Setter
  public void setCatColor(String catColor) {
     this.catColor = catColor;
  }
  // toString Method
  @Override
  public String toString() {
     return super.toString() + ", CatColor=" + catColor;
  }
}
```

==========

PetShelter.java:

```
package entity;
import dao.IAdoptable;
import exception.FileReadException;
```

```java
import util.DBConnUtil;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class PetShelter implements IAdoptable {
  private List<Pet> availablePets;
  // Constructor
  public PetShelter() {
    availablePets = new ArrayList<>();
  }
  // Add pet to the shelter
  public void addPet(Pet pet) {
    availablePets.add(pet);
  }
  // Remove pet from the shelter
  public void removePet(Pet pet) {
    availablePets.remove(pet);
  }
  // List all available pets
  public void listAvailablePets() {
    if (availablePets.isEmpty()) {
      System.out.println("No pets available for adoption.");
    } else {
      for (Pet pet : availablePets) {
        try {
          if (pet == null || pet.getName() == null || pet.getAge() <= 0) {
            throw new NullPointerException("Pet info is missing or invalid.");
          }
          System.out.println(pet.toString());
        } catch (NullPointerException e) {
          System.out.println("Error: " + e.getMessage());
        }
      }
    }
  }

  public List<Pet> getAvailablePets() {
    return availablePets;
  }
```

```java
  public void fetchPetsFromDatabase() throws FileReadException {
      Connection conn = DBConnUtil.getConnection("db.properties");
      if (conn == null) throw new FileReadException("Failed to connect to DB");
      String query = "SELECT * FROM pets";
      try (PreparedStatement stmt = conn.prepareStatement(query); ResultSet rs =
stmt.executeQuery()) {
          availablePets.clear();
          while (rs.next()) {
              String name = rs.getString("name");
              int age = rs.getInt("age");
              String breed = rs.getString("breed");
              String type = rs.getString("type");
              String catColor = rs.getString("cat_color");
              String dogBreed = rs.getString("dog_breed");
              Pet pet = null;
              if ("Cat".equalsIgnoreCase(type)) {
                  pet = new Cat(name, age, breed, catColor);
              } else if ("Dog".equalsIgnoreCase(type)) {
                  pet = new Dog(name, age, breed, dogBreed);
              }
              if (pet != null) {
                  addPet(pet);
              }
          }
      } catch (SQLException e) {
          throw new FileReadException("Error reading pets from DB: " + e.getMessage());
      }
  }
  // Implementation of adopt() from IAdoptable
  @Override
  public void adopt() {
      System.out.println("Pet adopted from the shelter.");
  }
}
```

==========

Exception package:

AdoptionException.java

```
package exception;
public class AdoptionException extends Exception {
  public AdoptionException(String message) {
      super(message);
  }
}
```

==========

FileReadException.java:

```
package exception;
public class FileReadException extends Exception {
  public FileReadException(String message) {
      super(message);
  }
}
```

==========

InsufficientFundsException.java:

```
package exception;
public class InsufficientFundsException extends Exception {
  public InsufficientFundsException(String message) {
      super(message);
  }
}
```

==========

InvalidAgeException.java:

```
package exception;
public class InvalidAgeException extends Exception {
```

```
  public InvalidAgeException(String message) {
      super(message);
  }
}
```

==========

NullPropertyException.java

```
package exception;
public class NullPropertyException extends Exception {
  public NullPropertyException(String message) {
      super(message);
  }
}
```

==========

Main package:

MainModule.java:

```
package main;
import java.util.Scanner;
import entity.*;
import dao.*;
import exception.*;
import java.io.*;
import java.sql.Connection;
import java.util.List;
import util.DBConnUtil;
public class MainModule {
  public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      PetShelter shelter = new PetShelter();
      AdoptionEvent event = new AdoptionEvent();
      String dbFile = "db.properties";
      // DAO Instances
      PetDAO petDAO = new PetDAO();
      CashDonationDAO cashDonationDAO = new CashDonationDAO();
```

```java
ItemDonationDAO itemDonationDAO = new ItemDonationDAO();
AdoptionEventDAO eventDAO = new AdoptionEventDAO();
ParticipantDAO participantDAO = new ParticipantDAO();
while (true) {
    System.out.println("\n===== PetPals Adoption Platform =====");
    System.out.println("1. Add a Pet");
    System.out.println("2. List Available Pets");
    System.out.println("3. Remove a Pet");
    System.out.println("4. Make a Cash Donation");
    System.out.println("5. Make an Item Donation");
    System.out.println("6. Host Adoption Event");
    System.out.println("7. Register Participant");
    System.out.println("8. Read Pets from File");
    System.out.println("9. Exit");
    System.out.print("Enter choice: ");
    int choice = scanner.nextInt();
    scanner.nextLine();
    try {
        switch (choice) {
            case 1:
                System.out.print("Enter pet type (dog/cat): ");
                String type = scanner.nextLine().toLowerCase();
                System.out.print("Enter name: ");
                String name = scanner.nextLine();
                System.out.print("Enter age: ");
                int age = scanner.nextInt();
                scanner.nextLine();
                if (age <= 0) {
                    throw new InvalidAgeException("Pet age must be a positive integer.");
                }
                System.out.print("Enter breed: ");
                String breed = scanner.nextLine();
                Pet pet = null;
                if (type.equals("dog")) {
                    System.out.print("Enter dog breed: ");
                    String dogBreed = scanner.nextLine();
                    pet = new Dog(name, age, breed, dogBreed);
                } else if (type.equals("cat")) {
                    System.out.print("Enter cat color: ");
                    String catColor = scanner.nextLine();
```

```
                    pet = new Cat(name, age, breed, catColor);
                } else {
                    System.out.println("Invalid pet type.");
                    break;
                }
                petDAO.addPet(pet);
                shelter.addPet(pet);
                System.out.println("Pet added successfully to both system and
database.");
                break;
            case 2:
                List<Pet> pets = petDAO.getAllPets();
                if (pets.isEmpty()) {
                    System.out.println("No pets available in the database.");
                } else {
                    System.out.println("Available Pets:");
                    for (Pet p : pets) {
                        if (p.getName() == null || p.getAge() == 0) {
                            throw new NullPropertyException("Pet information is
incomplete.");
                        }
                        System.out.println(p);
                    }
                }
                break;
            case 3:
                System.out.print("Enter pet name to remove: ");
                String removeName = scanner.nextLine();

                List<Pet> petsInDb = petDAO.getAllPets();
                Pet petToRemove = null;
                for (Pet p : petsInDb) {
                    if (p.getName().equalsIgnoreCase(removeName)) {
                        petToRemove = p;
                        break;
                    }
                }
                if (petToRemove != null) {
                    petDAO.removePetByName(removeName);
                    shelter.removePet(petToRemove); // In case it's in memory
```

```
                System.out.println("Pet '" + removeName + "' removed from
database.");
            } else {
                throw new AdoptionException("Pet not found.");
            }
            break;
        case 4:
            System.out.print("Enter donor name: ");
            String donorName = scanner.nextLine();
            System.out.print("Enter donation amount: ");
            double amount = scanner.nextDouble();
            scanner.nextLine();
            if (amount < 10) {
                throw new InsufficientFundsException("Minimum donation amount is
₹10.");
            }
            cashDonationDAO.addDonation(donorName, amount);
            System.out.println("Cash donation recorded in database.");
            break;
        case 5:
            System.out.print("Enter donor name: ");
            String donorItemName = scanner.nextLine();
            System.out.print("Enter item type: ");
            String item = scanner.nextLine();
            itemDonationDAO.addDonation(donorItemName, item);
            System.out.println("Item donation recorded in database.");
            break;
        case 6:
            System.out.print("Enter event description: ");
            String eventDesc = scanner.nextLine();
            System.out.print("Enter event location: ");
            String eventLocation = scanner.nextLine();
            int eventId = eventDAO.hostEvent(eventDesc, eventLocation);
            for (IAdoptable participant : event.getParticipants()) {
                if (participant instanceof PetShelter) {
                    participantDAO.addParticipant(eventId, "Pet Shelter");
                }
            }
            event.hostEvent();
            break;
```

```
                case 7:
                    System.out.print("Enter Event ID to register for: ");
                    int eventIdManual = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Enter Shelter Name to register: ");
                    String shelterName = scanner.nextLine();
                    participantDAO.addParticipant(eventIdManual, shelterName);
                    System.out.println("Shelter registered for the adoption event in
database.");
                    break;
                case 8:
                    System.out.print("Enter file name to read pets: ");
                    String fileName = scanner.nextLine();
                    try (BufferedReader br = new BufferedReader(new
FileReader(fileName))) {
                        String line;
                        while ((line = br.readLine()) != null) {
                            System.out.println("File Line: " + line);
                        }
                    } catch (IOException e) {
                        throw new FileReadException("Error reading file: " + e.getMessage());
                    }
                    break;
                case 9:
                    System.out.println("Exiting... Thank you for using PetPals.");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please select a valid option.");
            }
        } catch (InvalidAgeException | NullPropertyException |
InsufficientFundsException |
                AdoptionException | FileReadException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
  }
}
```

==========

TestConnection.java:

```
package main;
import java.sql.Connection;
import util.DBConnUtil;
public class TestConnection {
  public static void main(String[] args) {
      Connection conn = DBConnUtil.getConnection("db.properties");
      if (conn != null) {
          System.out.println("Database connection successful.");
      } else {
          System.out.println("Failed to connect to database.");
      }
  }
}
```

==========

util.package :

DBConnUtil .java

```
package util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConnUtil {
  public static Connection getConnection(String propertyFileName) {
      Connection conn = null;
      try {
          String connectionString =
DBPropertyUtil.getConnectionString(propertyFileName);
          if (connectionString != null) {
              conn = DriverManager.getConnection(connectionString);
          }
```

```
    } catch (SQLException e) {
        System.out.println("Database connection failed: " + e.getMessage());
    }
    return conn;
  }
}
```

==========

DBPropertyUtil.java:

```
package util;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
public class DBPropertyUtil {
  public static String getConnectionString(String propertyFileName) {
    Properties prop = new Properties();
    try (FileInputStream input = new FileInputStream(propertyFileName)) {
        prop.load(input);
        String url = prop.getProperty("db.url");
        String username = prop.getProperty("db.username");
        String password = prop.getProperty("db.password");
        return url + "?user=" + username + "&password=" + password;
    } catch (IOException e) {
        System.out.println("Error reading property file: " + e.getMessage());
        return null;
    }
  }
}
```

==========

db.properties:

db.url=jdbc:mysql://localhost:3306/petpalsdb
db.username=root
db.password=

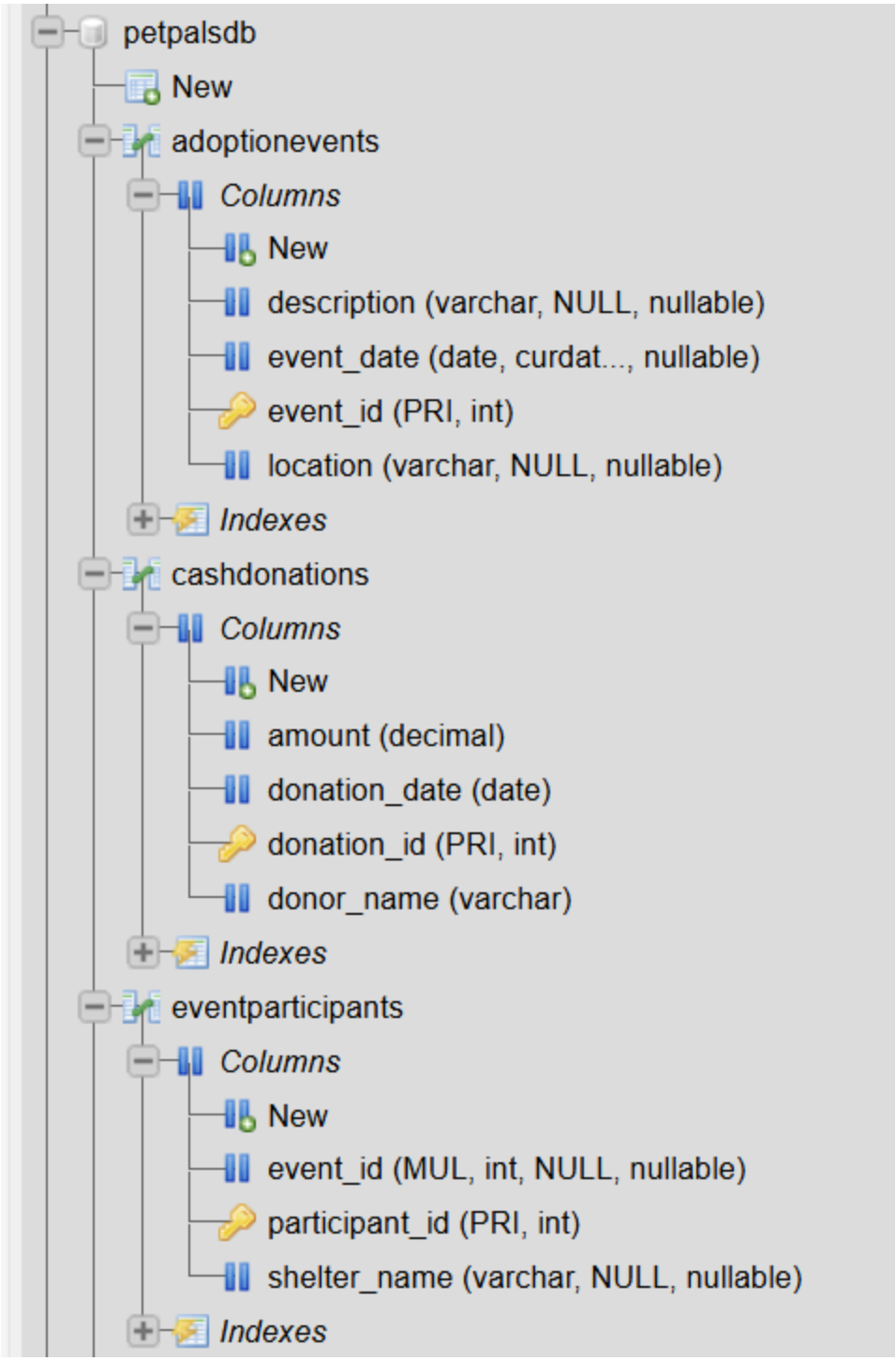
==========

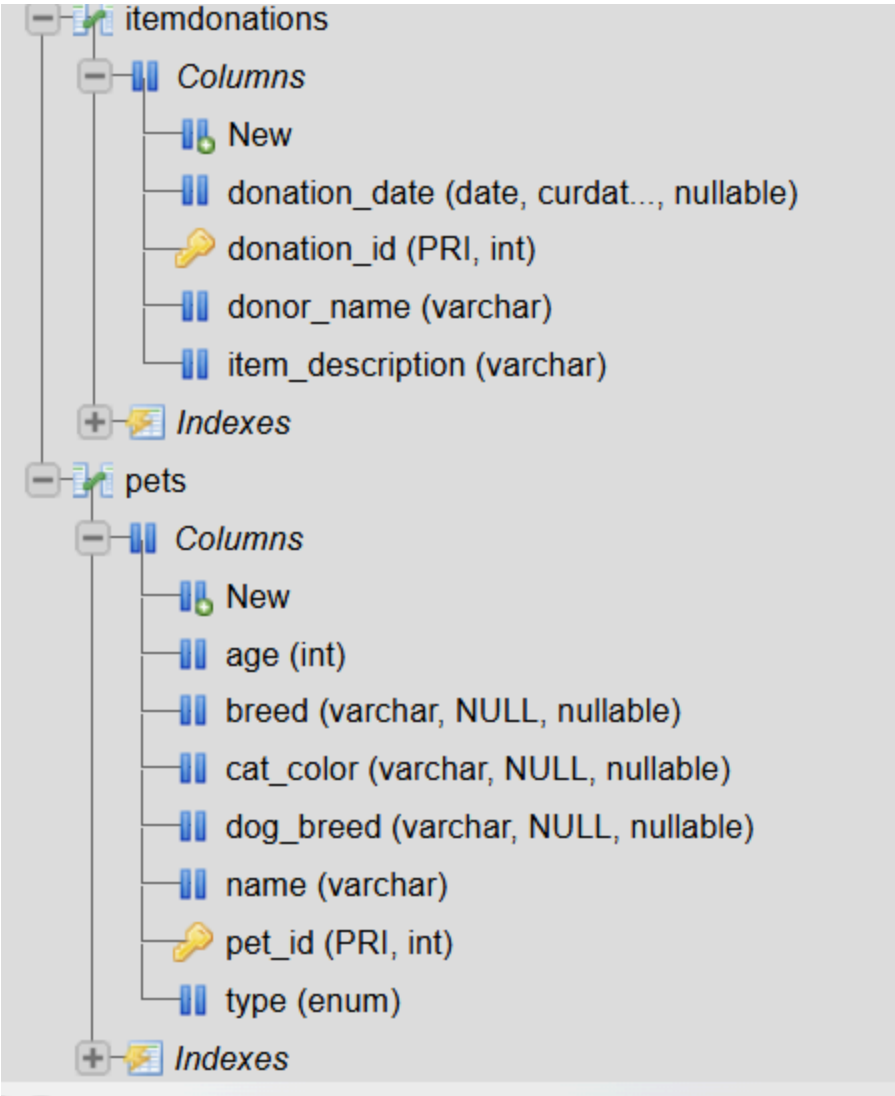
pets.txt:

Dog,Bruno,3,Labrador,Golden
Cat,Misty,2,Persian,White
Dog,Rocky,4,Beagle,Brown

==========

DB structure:


petpalsdb
New
adoptionevents
Columns
New
description (varchar, NULL, nullable)
event_date (date, curdat..., nullable)
event_id (PRI, int)
location (varchar, NULL, nullable)
Indexes
cashdonations
Columns
New
amount (decimal)
donation_date (date)
donation_id (PRI, int)
donor_name (varchar)
Indexes
eventparticipants
Columns
New
event_id (MUL, int, NULL, nullable)
participant_id (PRI, int)
shelter_name (varchar, NULL, nullable)
Indexes


itemdonations
Columns
New
donation_date (date, curdat..., nullable)
donation_id (PRI, int)
donor_name (varchar)
item_description (varchar)
Indexes
pets
Columns
New
age (int)
breed (varchar, NULL, nullable)
cat_color (varchar, NULL, nullable)
dog_breed (varchar, NULL, nullable)
name (varchar)
pet_id (PRI, int)
type (enum)
Indexes

Output:

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 1
Enter pet type (dog/cat): dog
Enter name: moti
Enter age: 3
Enter breed: Labrador
Enter dog breed: Labrador
☑ Pet added to the database.
Pet added successfully to both system and database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 2
Available Pets:
Pet [Name=Tommy, Age=3, Breed=Labrador], DogBreed=Golden Labrador
Pet [Name=Bruno, Age=5, Breed=Beagle], DogBreed=Beagle
Pet [Name=Mani, Age=3, Breed=indi], CatColor=white
Pet [Name=moti, Age=3, Breed=Labrador], DogBreed=Labrador
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 1
Enter pet type (dog/cat): cat
Enter name: mani
Enter age: 2
Enter breed: indi
Enter cat color: brown
☑ Pet added to the database.
Pet added successfully to both system and database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 2
Available Pets:
Pet [Name=Tommy, Age=3, Breed=Labrador], DogBreed=Golden Labrador
Pet [Name=Bruno, Age=5, Breed=Beagle], DogBreed=Beagle
Pet [Name=Mani, Age=3, Breed=indi], CatColor=white
Pet [Name=moti, Age=3, Breed=Labrador], DogBreed=Labrador
Pet [Name=mani, Age=2, Breed=indi], CatColor=brown
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 3
Enter pet name to remove: tommy
Error: Pet not found.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 3
Enter pet name to remove: mani
Pet removed from the database.
Pet 'mani' removed from database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 4
Enter donor name: Sanchita
Enter donation amount: 4000
Cash donation recorded in database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 5
Enter donor name: Ram
Enter item type: Medicine
 Item donation inserted in database.
Item donation recorded in database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 6
Enter event description: Save animal
Enter event location: Pune
Adoption event recorded in DB with ID: 7
Hosting Adoption Event...
Adoption Event concluded.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 7
Enter Event ID to register for: 7
Enter Shelter Name to register: Animal Safety
Shelter registered for the adoption event in database.
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 8
Enter file name to read pets: pets.txt
File Line: Dog,Bruno,3,Labrador,Golden
File Line: Cat,Misty,2,Persian,White
File Line: Dog,Rocky,4,Beagle,Brown
```

```
===== PetPals Adoption Platform =====
1. Add a Pet
2. List Available Pets
3. Remove a Pet
4. Make a Cash Donation
5. Make an Item Donation
6. Host Adoption Event
7. Register Participant
8. Read Pets from File
9. Exit
Enter choice: 9
Exiting... Thank you for using PetPals.
```

Tables after Insertions:

Pets

Extra options

	pet_id	name	age	breed	type	dog_breed	cat_color
Edit Copy Delete	1	Tommy	3	Labrador	Dog	Golden Labrador	NULL
Edit Copy Delete	2	Bruno	5	Beagle	Dog	Beagle	NULL
Edit Copy Delete	10	moti	3	Labrador	Dog	Labrador	NULL

Itemdonations:

Extra options

	donation_id	donor_name	item_description	donation_date
Edit Copy Delete	1	Sneha Kapoor	Pack of dog food	2025-04-09
Edit Copy Delete	2	Amit Kumar	Cat toys and grooming kit	2025-04-09
Edit Copy Delete	6	Ram	Medicine	2025-04-09

eventparticipants

Extra options

	participant_id	event_id	shelter_name
Edit Copy Delete	2	1	Animal Rescue League
Edit Copy Delete	4	1	sahas
Edit Copy Delete	6	7	Animal Safety

cashdonations

←T→	donation_id	donor_name	amount	donation_date
Edit Copy Delete	4	Madhu	4777.00	2025-04-09
Edit Copy Delete	7	diksha	4770.00	2025-04-09
Edit Copy Delete	9	Sanchita	4000.00	2025-04-09

Adoptionevents

←T→	event_id	event_date	description	location
Edit Copy Delete	1	2025-04-10	Spring Pet Adoption Fair	*NULL*
Edit Copy Delete	3	2025-04-09	Monthly Adoption Event	*NULL*
Edit Copy Delete	7	2025-04-09	Save animal	Pune

Github Repo:

https://github.com/SanchitaDevkar/Petpals.git