

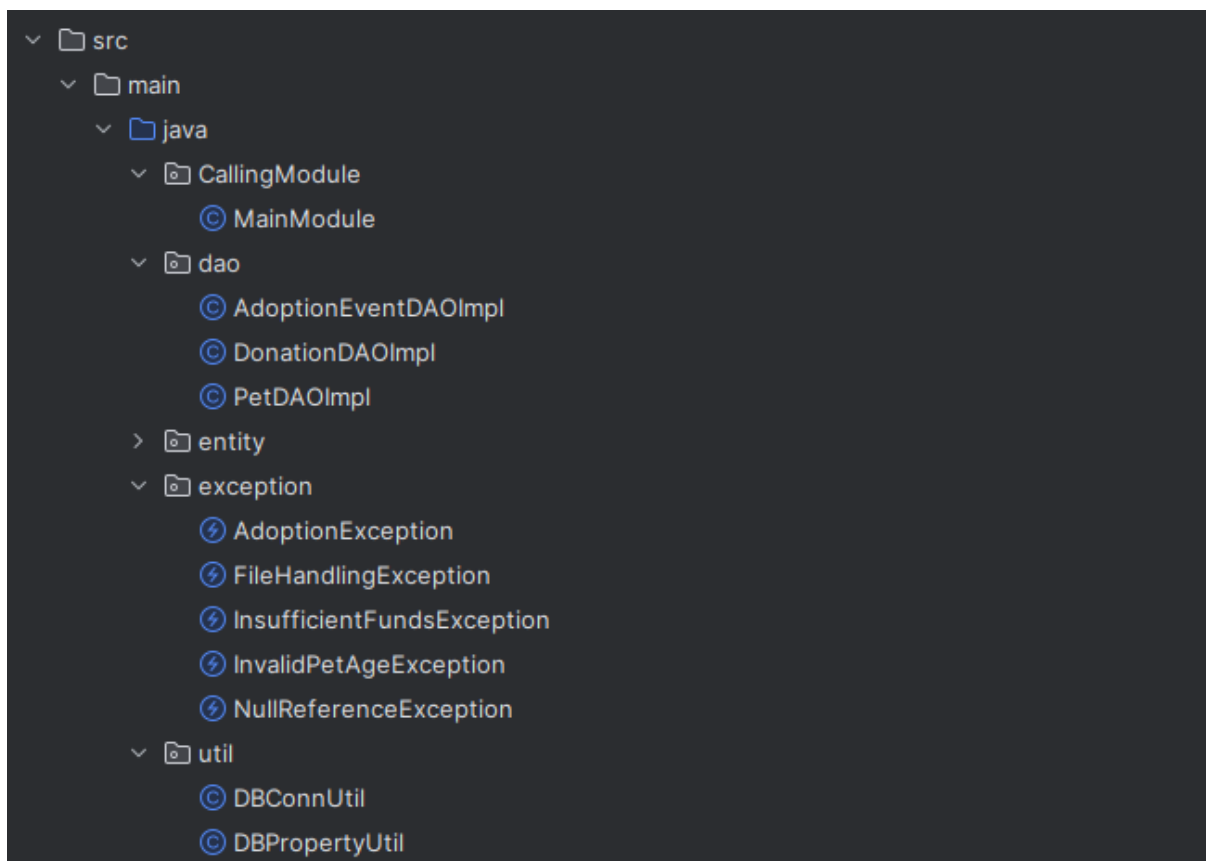
Java coding challenge – Pet Pals

Melvin Jones

Key Features

- Listing pets for adoption.
- Managing donations for shelters.
- Hosting and managing adoption events.

My project Structure :



Created SQL Schema for the tables mentioned in the document

```
+-----+
| Tables_in_javacodingchallenge |
+-----+
| adoption_events                |
| adoptionevents                 |
| donations                      |
| participants                   |
| pets                           |
+-----+

5 rows in set (0.04 sec)
```

Inserted values to the tables, such as pets, donations, etc..

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'application' selected. The main window shows a SQL query in the 'SQL File 8' tab:

```
1 use javacodingchallenge;
2 show tables;
3 select * from pets;
4
5
6
```

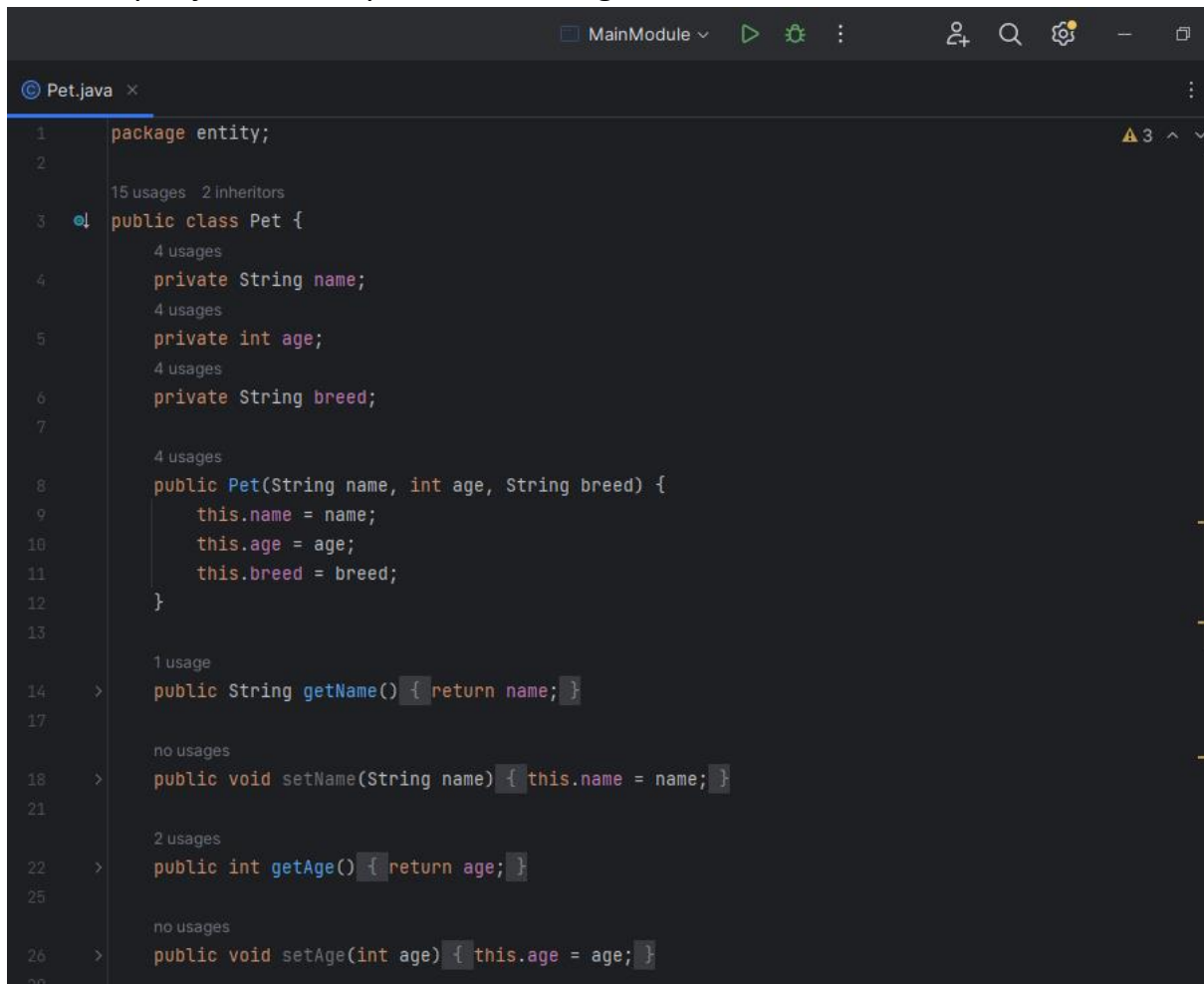
The 'Result Grid' at the bottom displays the results of the query:

	id	name	age	breed	type	specific_detail
▶	1	Buddy	3	Golden Retriever	Dog	Golden
	2	Whiskers	2	Siberian	Cat	Gray
•	NULL	NULL	NULL	NULL	NULL	NULL

The bottom status bar indicates 'Schema: application'.

ENTITY PACKAGE :

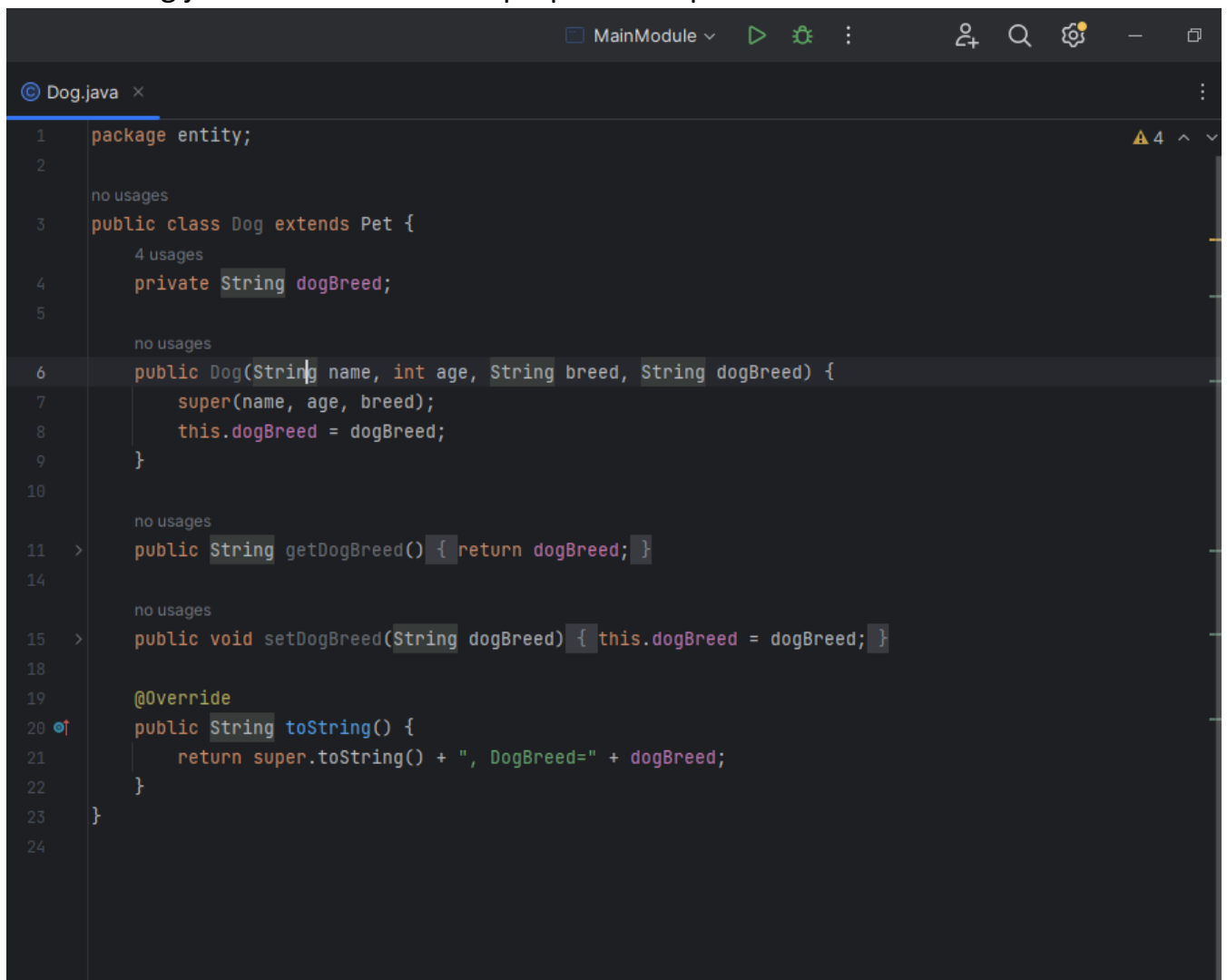
Created pet.java and implemented the getter setter methods, for it



```
1 package entity;
2
3 public class Pet {
4     private String name;
5     private int age;
6     private String breed;
7
8     public Pet(String name, int age, String breed) {
9         this.name = name;
10        this.age = age;
11        this.breed = breed;
12    }
13
14    public String getName() { return name; }
15
16    public void setName(String name) { this.name = name; }
17
18    public int getAge() { return age; }
19
20    public void setAge(int age) { this.age = age; }
```

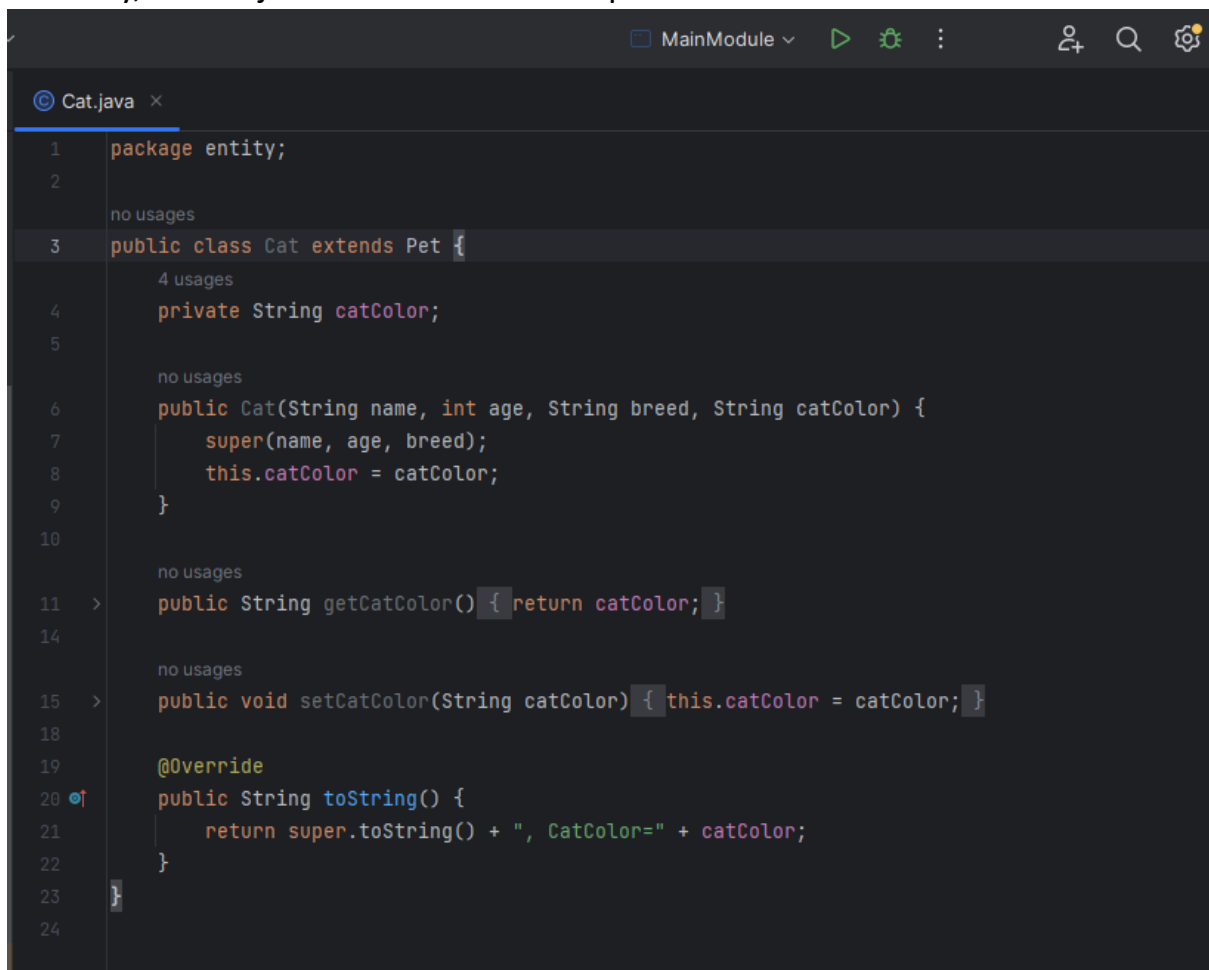
The screenshot shows an IDE window titled 'Pet.java'. The code defines a 'Pet' class in the 'entity' package. It has three private attributes: 'name' (String), 'age' (int), and 'breed' (String). The class includes a constructor 'Pet(String name, int age, String breed)' that initializes these attributes. It also has four public methods: 'getName()' returning the name, 'setName(String name)' setting the name, 'getAge()' returning the age, and 'setAge(int age)' setting the age. The IDE interface includes a top bar with icons for running, debugging, and other functions, and a left sidebar with a file explorer.

Created Dog.java which inherits the properties of pet class



```
1 package entity;
2
3 public class Dog extends Pet {
4     private String dogBreed;
5
6     public Dog(String name, int age, String breed, String dogBreed) {
7         super(name, age, breed);
8         this.dogBreed = dogBreed;
9     }
10
11     public String getDogBreed() { return dogBreed; }
12
13     public void setDogBreed(String dogBreed) { this.dogBreed = dogBreed; }
14
15     @Override
16     public String toString() {
17         return super.toString() + ", DogBreed=" + dogBreed;
18     }
19 }
```

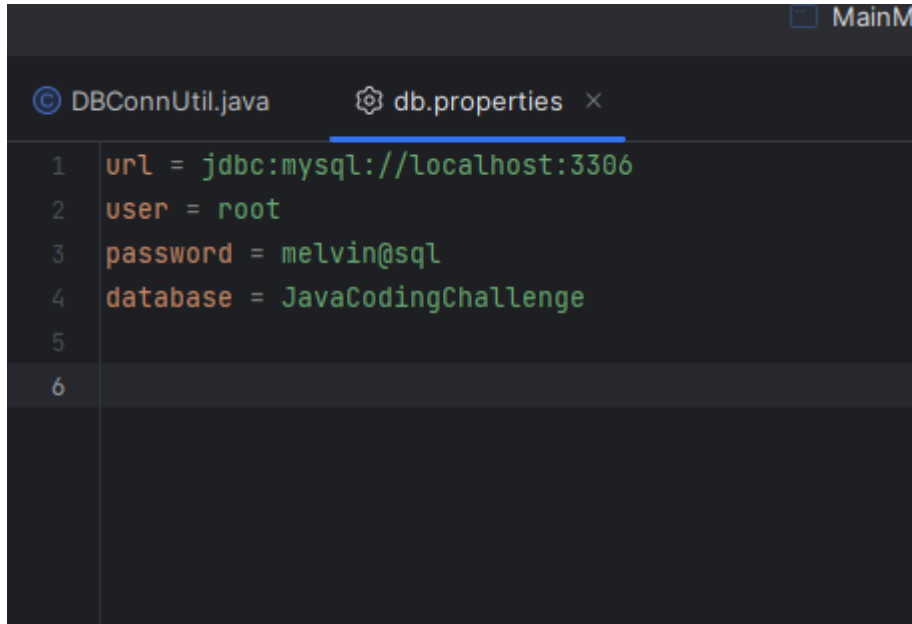
Similarly, for cat.java which inherits the pet class



```
1 package entity;
2
3 public class Cat extends Pet {
4     private String catColor;
5
6     public Cat(String name, int age, String breed, String catColor) {
7         super(name, age, breed);
8         this.catColor = catColor;
9     }
10
11     public String getCatColor() { return catColor; }
12
13
14
15     public void setCatColor(String catColor) { this.catColor = catColor; }
16
17
18
19     @Override
20     public String toString() {
21         return super.toString() + ", CatColor=" + catColor;
22     }
23 }
24
```

UTIL PACKAGE

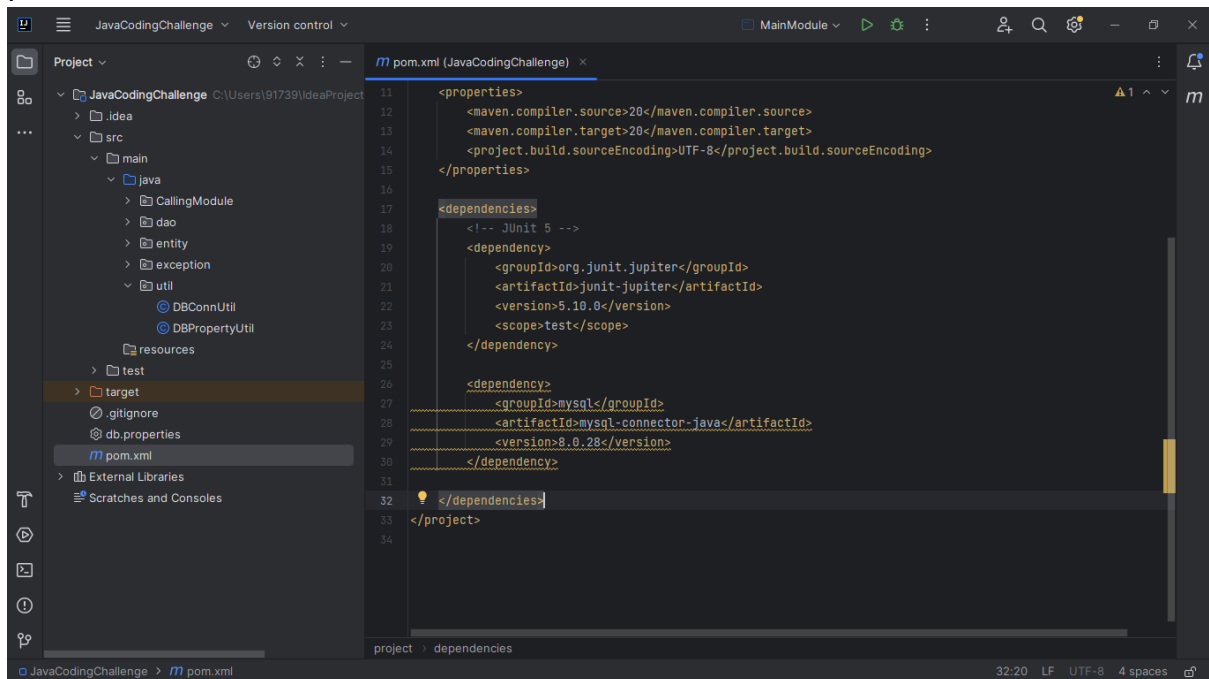
Next created Util package to make database connection , then created db.properties file



The screenshot shows an IDE window with two tabs: 'DBConnUtil.java' and 'db.properties'. The 'db.properties' tab is active, displaying the following configuration:

```
1 url = jdbc:mysql://localhost:3306
2 user = root
3 password = melvin@sql
4 database = JavaCodingChallenge
5
6
```

Added Mysql connector with the help of maven , added dependency in pom.xml file



The screenshot shows an IDE window with the 'pom.xml (JavaCodingChallenge)' file open. The file content is as follows:

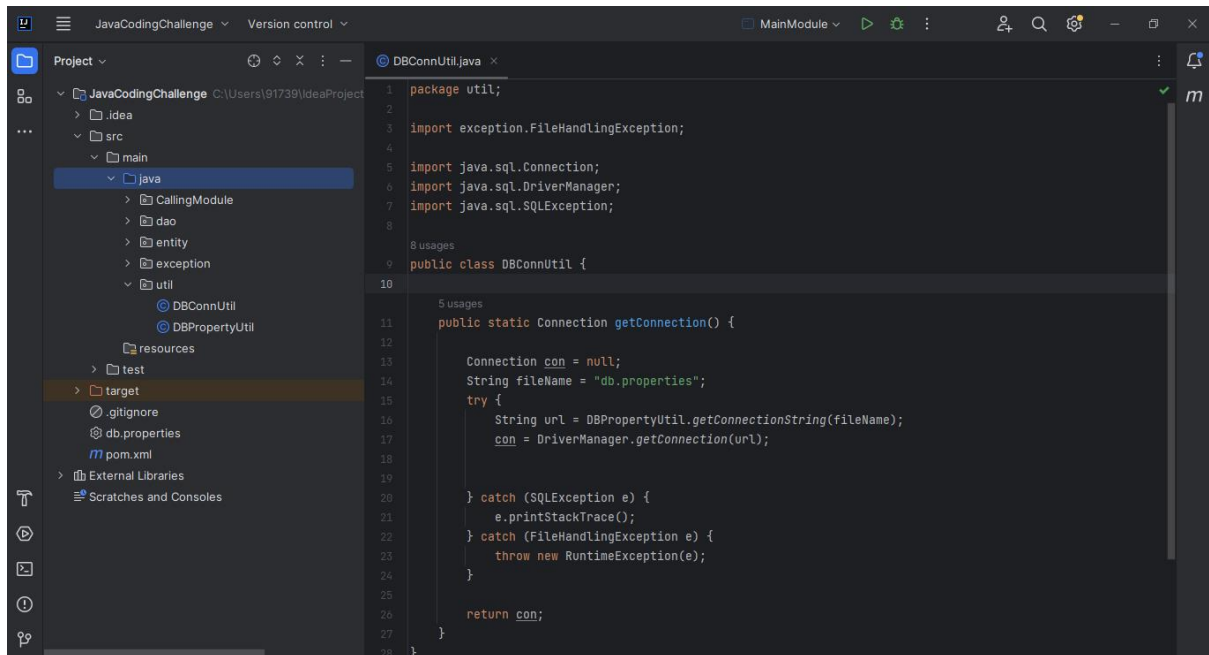
```
11 <properties>
12   <maven.compiler.source>20</maven.compiler.source>
13   <maven.compiler.target>20</maven.compiler.target>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15 </properties>
16
17 <dependencies>
18   <!-- JUnit 5 -->
19   <dependency>
20     <groupId>org.junit.jupiter</groupId>
21     <artifactId>junit-jupiter</artifactId>
22     <version>5.10.0</version>
23     <scope>test</scope>
24   </dependency>
25
26   <dependency>
27     <groupId>mysql</groupId>
28     <artifactId>mysql-connector-java</artifactId>
29     <version>8.0.28</version>
30   </dependency>
31
32 </dependencies>
33 </project>
34
```

The IDE interface also shows a project tree on the left with the following structure:

- JavaCodingChallenge
 - .idea
 - src
 - main
 - java
 - CallingModule
 - dao
 - entity
 - exception
 - util
 - DBConnUtil
 - DBPropertyUtil
 - resources
 - test
 - target
 - .gitignore
 - db.properties
 - pom.xml
 - External Libraries
 - Scratches and Consoles

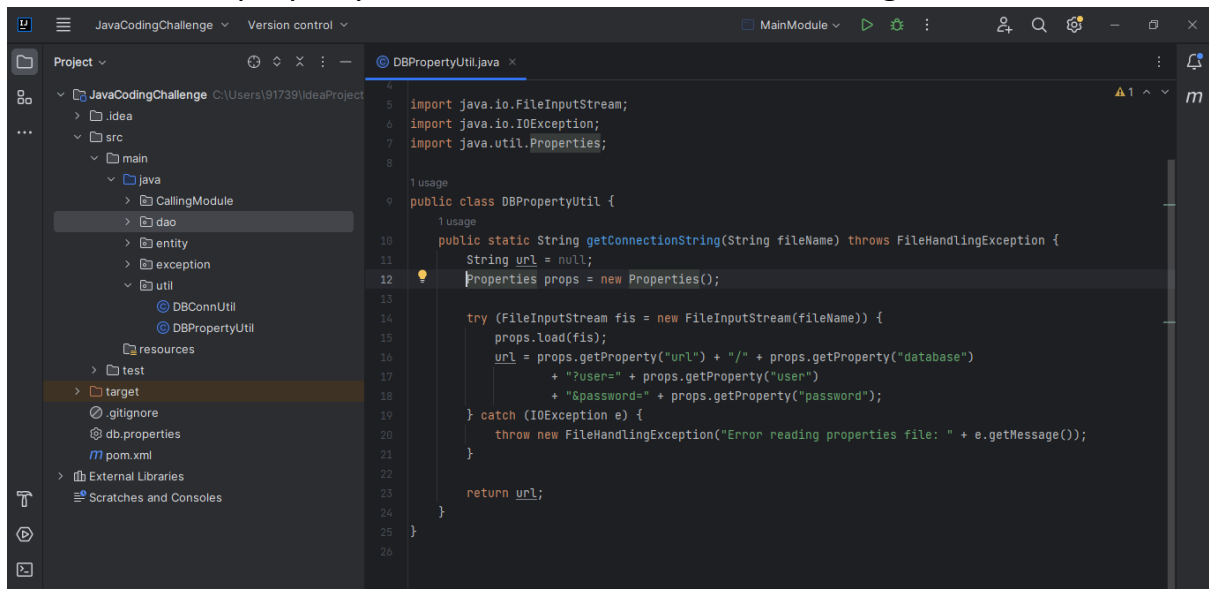
The status bar at the bottom indicates the file is at line 32, column 20, using LF line endings, UTF-8 encoding, and 4 spaces for indentation.

Created DBConnUtil.java class to establish a connection



```
1 package util;
2
3 import exception.FileHandlingException;
4
5 import java.sql.Connection;
6 import java.sql.DriverManager;
7 import java.sql.SQLException;
8
9 public class DBConnUtil {
10
11     5 usages
12     public static Connection getConnection() {
13
14         Connection con = null;
15         String fileName = "db.properties";
16         try {
17             String url = DBPropertyUtil.getConnectionString(fileName);
18             con = DriverManager.getConnection(url);
19
20         } catch (SQLException e) {
21             e.printStackTrace();
22         } catch (FileHandlingException e) {
23             throw new RuntimeException(e);
24         }
25
26         return con;
27     }
28 }
```

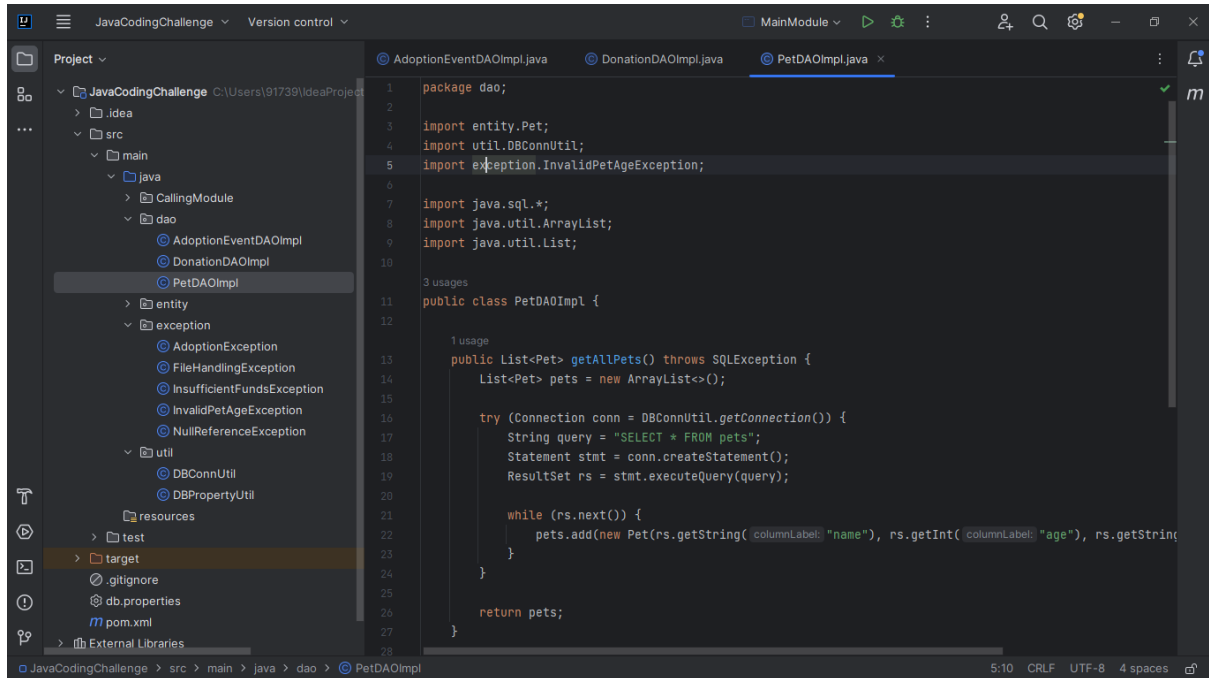
Created the DBproperty.util to create the connection string



```
4
5 import java.io.FileInputStream;
6 import java.io.IOException;
7 import java.util.Properties;
8
9 public class DBPropertyUtil {
10
11     1 usage
12     public static String getConnectionString(String fileName) throws FileHandlingException {
13
14         String url = null;
15         Properties props = new Properties();
16
17         try (FileInputStream fis = new FileInputStream(fileName)) {
18             props.load(fis);
19             url = props.getProperty("url") + "/" + props.getProperty("database")
20                 + "?user=" + props.getProperty("user")
21                 + "&password=" + props.getProperty("password");
22         } catch (IOException e) {
23             throw new FileHandlingException("Error reading properties file: " + e.getMessage());
24         }
25
26         return url;
27     }
28 }
```

DAO PACKAGE

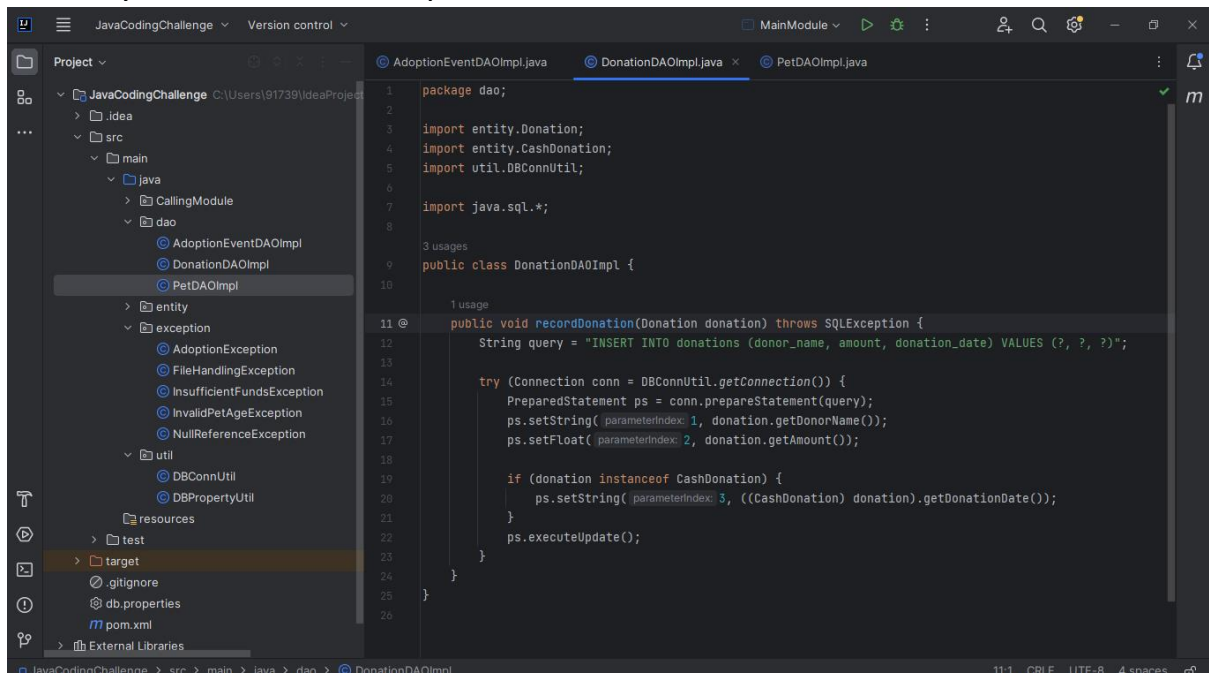
Created Dao package to implement the methods of List pets, Manage Donation, Manage Events



The screenshot shows the IntelliJ IDEA IDE with the 'PetDAOImpl.java' file open in the 'dao' package. The project structure on the left includes 'JavaCodingChallenge' with subdirectories like 'src', 'main', 'java', 'dao', 'entity', 'exception', and 'util'. The code in 'PetDAOImpl.java' is as follows:

```
1 package dao;
2
3 import entity.Pet;
4 import util.DBConnUtil;
5 import exception.InvalidPetAgeException;
6
7 import java.sql.*;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class PetDAOImpl {
12
13     public List<Pet> getAllPets() throws SQLException {
14         List<Pet> pets = new ArrayList<>();
15
16         try (Connection conn = DBConnUtil.getConnection()) {
17             String query = "SELECT * FROM pets";
18             Statement stmt = conn.createStatement();
19             ResultSet rs = stmt.executeQuery(query);
20
21             while (rs.next()) {
22                 pets.add(new Pet(rs.getString("name"), rs.getInt("age"), rs.getString("breed")));
23             }
24         }
25
26         return pets;
27     }
28 }
```

Similarly, for DonationDAImpl

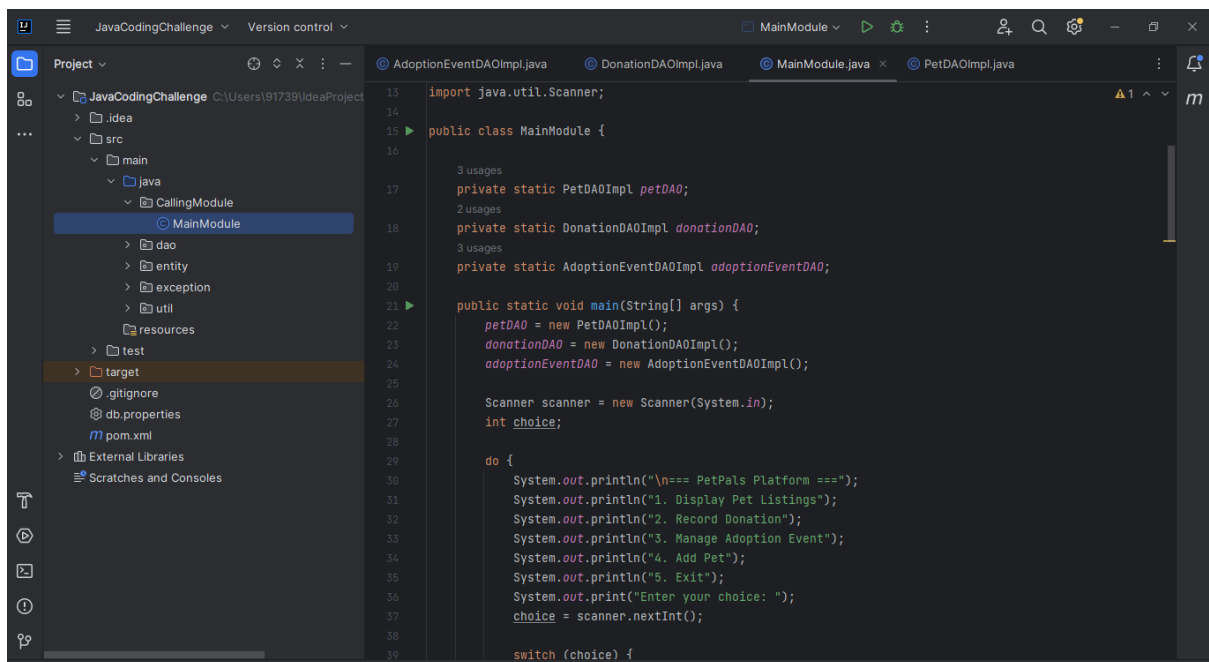


The screenshot shows the IntelliJ IDEA IDE with the 'DonationDAOImpl.java' file open in the 'dao' package. The project structure on the left is similar to the previous screenshot. The code in 'DonationDAOImpl.java' is as follows:

```
1 package dao;
2
3 import entity.Donation;
4 import entity.CashDonation;
5 import util.DBConnUtil;
6
7 import java.sql.*;
8
9 public class DonationDAOImpl {
10
11     public void recordDonation(Donation donation) throws SQLException {
12         String query = "INSERT INTO donations (donor_name, amount, donation_date) VALUES (?, ?, ?)";
13
14         try (Connection conn = DBConnUtil.getConnection()) {
15             PreparedStatement ps = conn.prepareStatement(query);
16             ps.setString(1, donation.getDonorName());
17             ps.setFloat(2, donation.getAmount());
18
19             if (donation instanceof CashDonation) {
20                 ps.setString(3, ((CashDonation) donation).getDonationDate());
21             }
22             ps.executeUpdate();
23         }
24     }
25 }
```


CALLINGMODULE PACKAGE

Next , Created the Main module to call the methods



```
import java.util.Scanner;

public class MainModule {

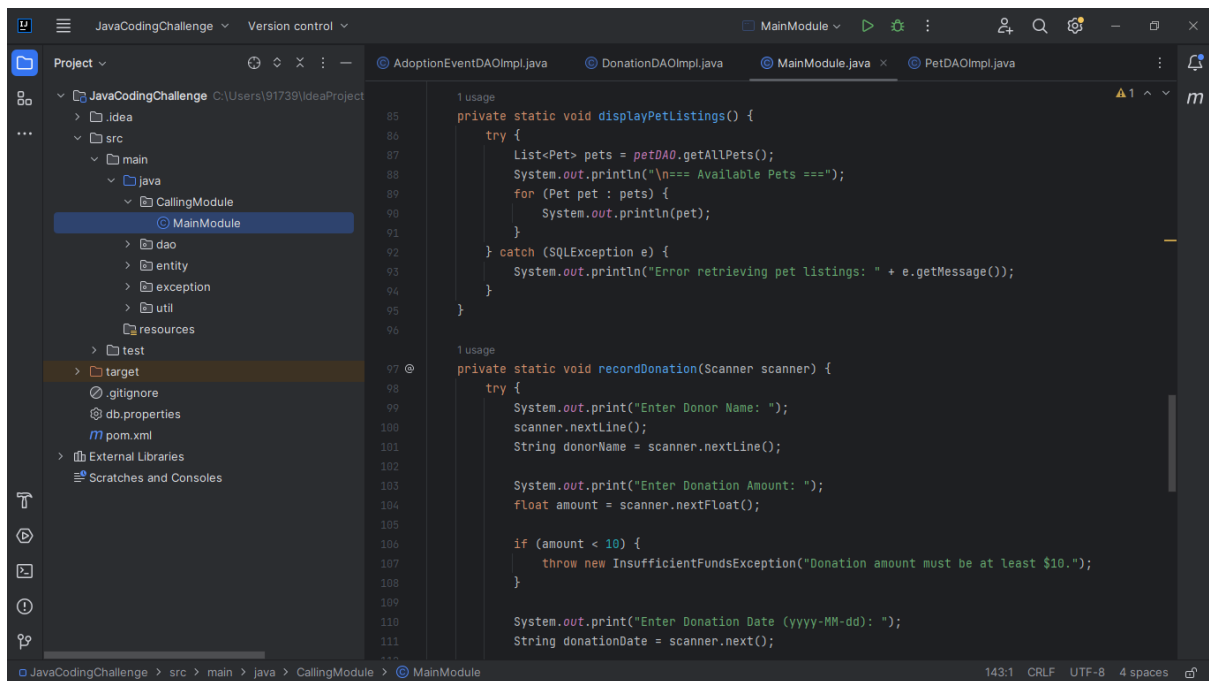
    3 usages
    private static PetDAOImpl petDAO;
    2 usages
    private static DonationDAOImpl donationDAO;
    3 usages
    private static AdoptionEventDAOImpl adoptionEventDAO;

    public static void main(String[] args) {
        petDAO = new PetDAOImpl();
        donationDAO = new DonationDAOImpl();
        adoptionEventDAO = new AdoptionEventDAOImpl();

        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n=== PetPals Platform ===");
            System.out.println("1. Display Pet Listings");
            System.out.println("2. Record Donation");
            System.out.println("3. Manage Adoption Event");
            System.out.println("4. Add Pet");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
```



```
1 usage
private static void displayPetListings() {
    try {
        List<Pet> pets = petDAO.getAllPets();
        System.out.println("\n=== Available Pets ===");
        for (Pet pet : pets) {
            System.out.println(pet);
        }
    } catch (SQLException e) {
        System.out.println("Error retrieving pet listings: " + e.getMessage());
    }
}

1 usage
private static void recordDonation(Scanner scanner) {
    try {
        System.out.print("Enter Donor Name: ");
        scanner.nextLine();
        String donorName = scanner.nextLine();

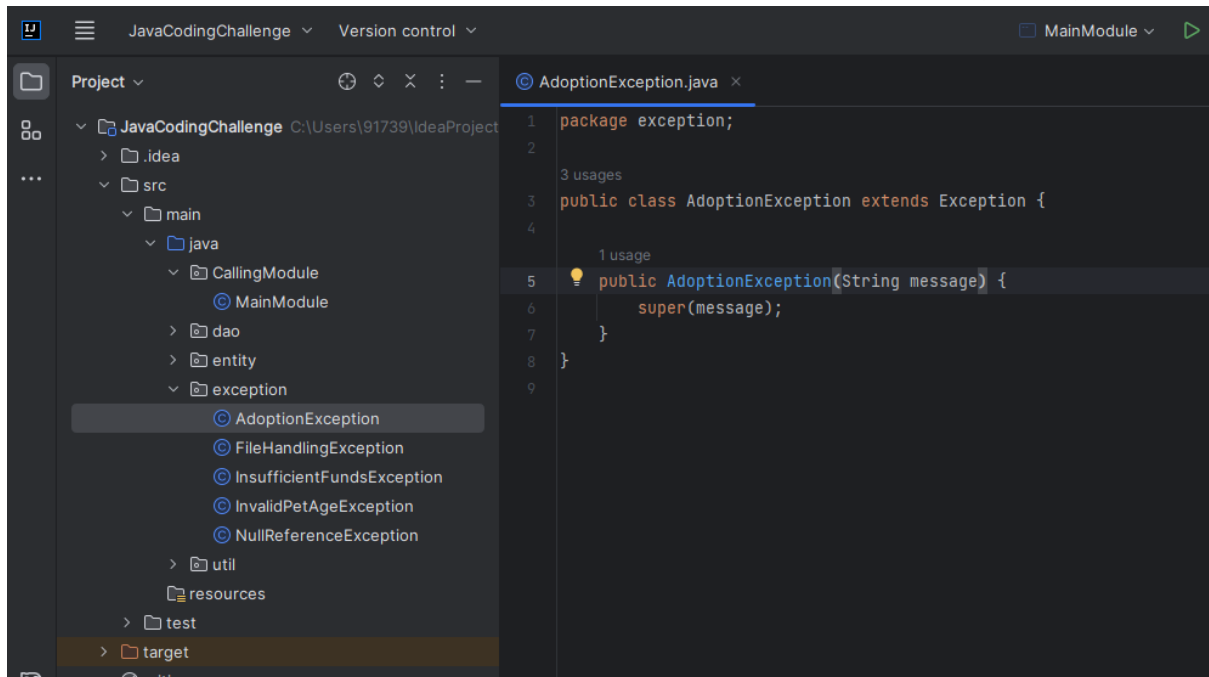
        System.out.print("Enter Donation Amount: ");
        float amount = scanner.nextFloat();

        if (amount < 10) {
            throw new InsufficientFundsException("Donation amount must be at least $10.");
        }

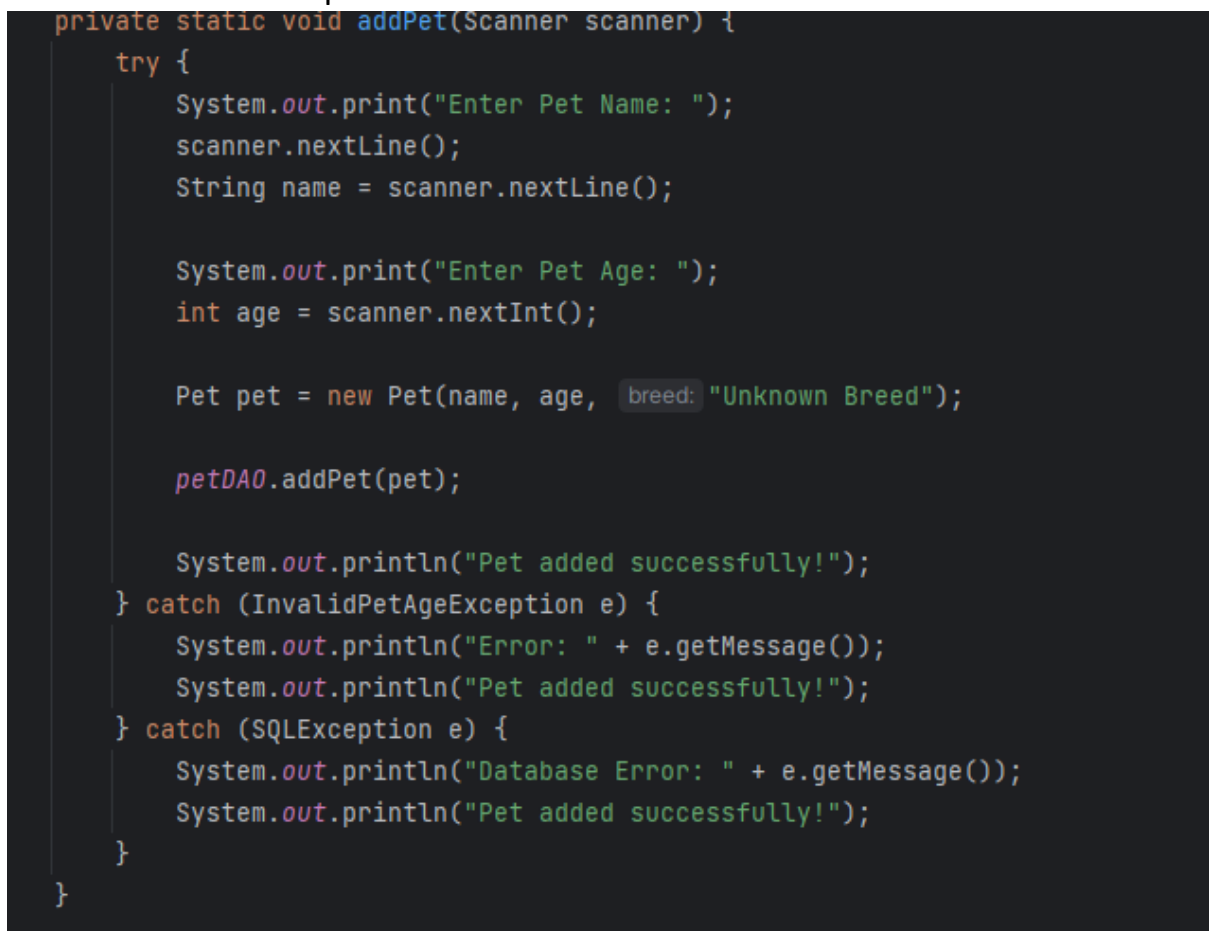
        System.out.print("Enter Donation Date (yyyy-MM-dd): ");
        String donationDate = scanner.next();
    }
}
```

EXCEPTIONS PACKAGE

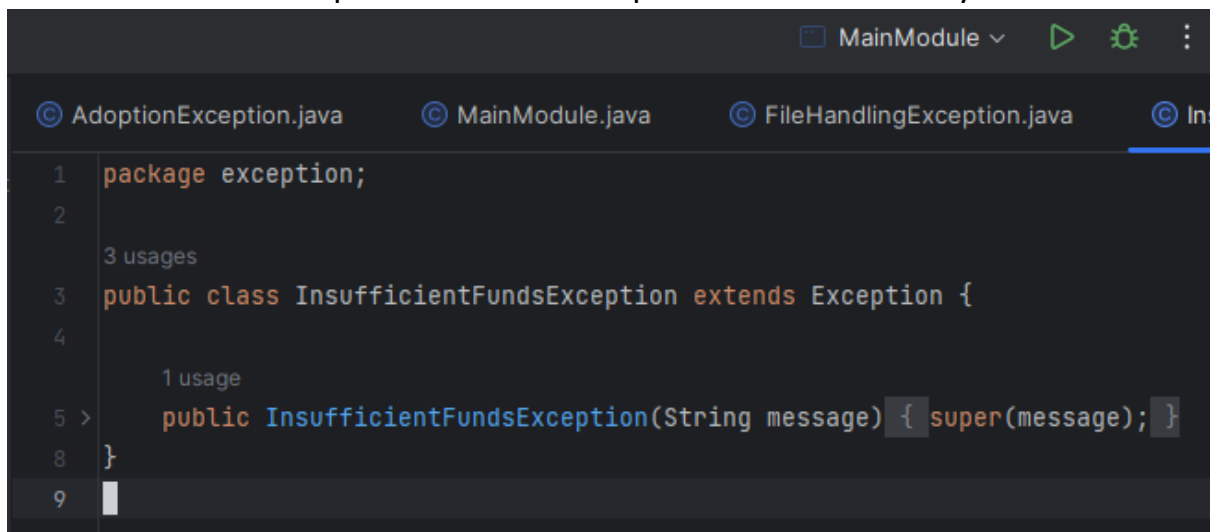
Then created exceptions for the required class in the exceptions package



Implemented the exceptions in Adoption method to check the age of pet if it is less than 0 then exception will be thrown



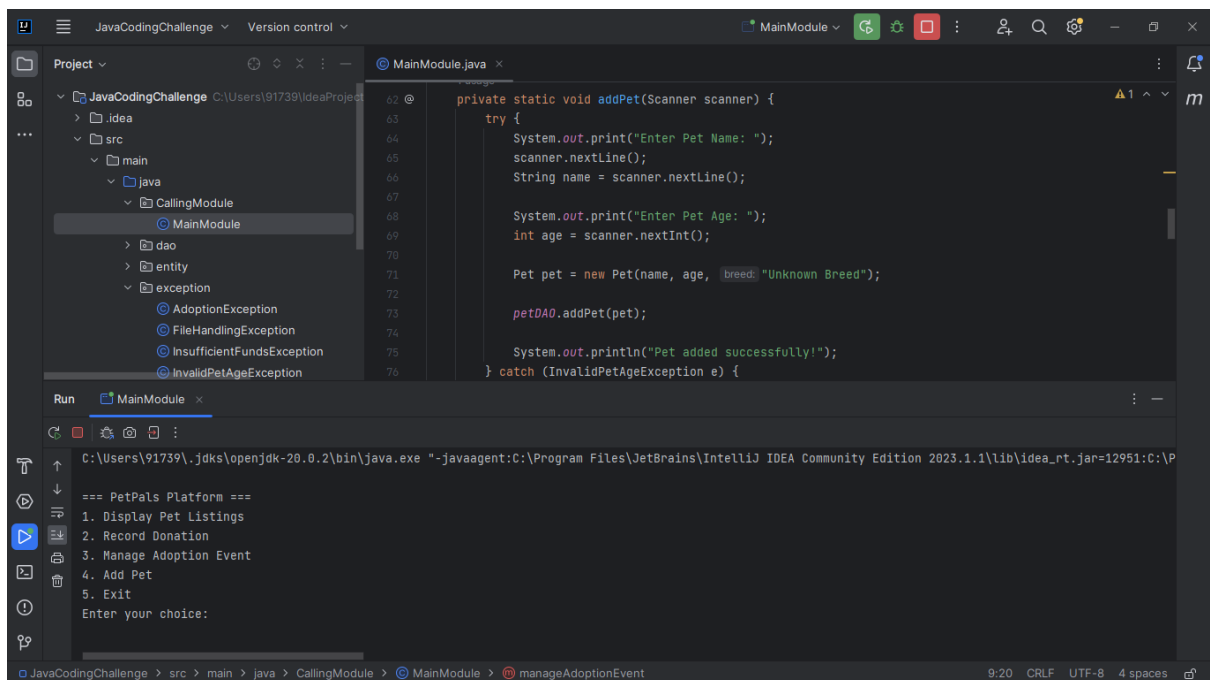
Likewise created exceptions for all the required methods one by one



```
1 package exception;
2
3 3 usages
4
5 > 1 usage
6 public class InsufficientFundsException extends Exception {
7
8     1 usage
9     public InsufficientFundsException(String message) { super(message); }
10
11 }
```

RESULTS

Finally executed the program and main method is triggered and the output is displayed



```
62 @
63
64 private static void addPet(Scanner scanner) {
65     try {
66         System.out.print("Enter Pet Name: ");
67         scanner.nextLine();
68         String name = scanner.nextLine();
69
70         System.out.print("Enter Pet Age: ");
71         int age = scanner.nextInt();
72
73         Pet pet = new Pet(name, age, breed: "Unknown Breed");
74
75         petDAO.addPet(pet);
76
77         System.out.println("Pet added successfully!");
78     } catch (InvalidPetAgeException e) {
```

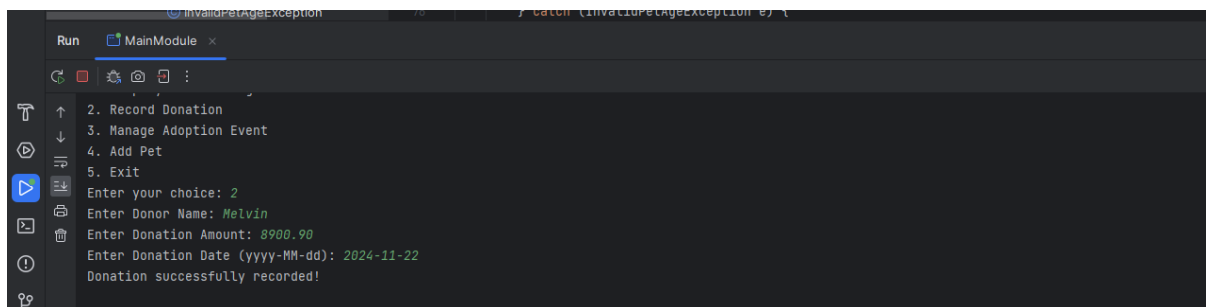
```
Run MainModule
C:\Users\91739\jdk\openjdk-20.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.1\lib\idea_rt.jar=12951:C:\P
=== PetPals Platform ===
1. Display Pet Listings
2. Record Donation
3. Manage Adoption Event
4. Add Pet
5. Exit
Enter your choice:
```

Selecting case 1

```
Enter your choice: 1

=== Available Pets ===
Pet [Name=Buddy, Age=3, Breed=Golden Retriever]
Pet [Name=Whiskers, Age=2, Breed=Siberian]
```

Selecting case 2



The screenshot shows a Java IDE with a 'Run' window open. The main code window displays a menu with five options: 1. Display Pet Listings, 2. Record Donation, 3. Manage Adoption Event, 4. Add Pet, and 5. Exit. The user has selected option 2. The program prompts for 'Enter Donor Name: Melvin', 'Enter Donation Amount: 8900.90', and 'Enter Donation Date (yyyy-MM-dd): 2024-11-22'. The final output is 'Donation successfully recorded!'. The 'Run' window shows the execution of 'MainModule'.

```
2. Record Donation
3. Manage Adoption Event
4. Add Pet
5. Exit
Enter your choice: 2
Enter Donor Name: Melvin
Enter Donation Amount: 8900.90
Enter Donation Date (yyyy-MM-dd): 2024-11-22
Donation successfully recorded!
```

Getting an exception if donation amount is less than 10



The screenshot shows a Java IDE with a 'Run' window open. The main code window displays the same menu as before. The user has selected option 2. The program prompts for 'Enter Donor Name: Jones' and 'Enter Donation Amount: 8'. The final output is 'Error: Donation amount must be at least \$10.'. The 'Run' window shows the execution of 'MainModule'.

```
=== PetPals Platform ===
1. Display Pet Listings
2. Record Donation
3. Manage Adoption Event
4. Add Pet
5. Exit
Enter your choice: 2
Enter Donor Name: Jones
Enter Donation Amount: 8
Error: Donation amount must be at least $10.
```

Case 5 to exit the program



The screenshot shows a Java IDE with a 'Run' window open. The main code window displays the same menu as before. The user has selected option 5. The program prompts for 'Enter your choice: 5' and the final output is 'Exiting...'. The 'Run' window shows the execution of 'MainModule'.

```
=== PetPals Platform ===
1. Display Pet Listings
2. Record Donation
3. Manage Adoption Event
4. Add Pet
5. Exit
Enter your choice: 5
Exiting...
```

Database getting updated after the output.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following queries:

```
1 • use javacodingchallenge;
2 • show tables;
3 • select * from donations;
4
5
```

The Results window displays the output of the third query, showing a table with 6 rows and 5 columns:

	id	donor_name	amount	donation_date	item_type
1	1	John Doe	100.00	2024-11-22	NULL
2	2	Jane Smith	50.00	2024-11-20	NULL
3	3	Melvin	7000.98	2024-01-04	NULL
4	4	Melvin	38000.00	2024-05-04	NULL
5	5	leo	10.00	2023-01-04	NULL
6	6	Melvin	8900.90	2024-11-22	NULL
•	NULL	NULL	NULL	NULL	NULL

The left sidebar shows the Schemas tree with the following structure:

- javacodingchallenge
 - Tables
 - adoption_events
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - adoptionevents
 - Columns
 - id
 - event_name
 - event_date
 - Indexes
 - Foreign Keys
 - Triggers
 - donations
 - participants

The Information window shows the details for the **Table: adoptionevents**:

Columns:

- id: int AI PK
- event_name: varchar(50)
- event_date: date

Finally, I have successfully implemented the methods and the Output is displayed as expected.