PROJECT NAME: Pet Adoption System

Purpose of this Project: This project aims to demonstrate object-oriented programming (OOP) concepts through a simple console based pet adoption system. It allows users to securely log in as Admin or Customer, manage pet listings, and handle adoption applications, showcasing encapsulation, inheritance, polymorphism, overloading, and overriding. By simulating a real world adoption process, it promotes efficient pet placement and animal welfare.

Name: Rezwan Hossain Prince

ID: 241400067

PROJECT CODE: NOTE: (Admin Pass: **Admin123**) (Customer Pass: **pet123**)

```
import java.util.ArrayList;

import java.util.Scanner;

abstract class User {

        private String name;

        private String password;

 public User(String name, String password) {

        this.name = name;

        this.password = password;

        }

 public String getName() {

        return name;

        }

public boolean checkPassword(String inputPassword) {
```

```
        return password.equals(inputPassword);
        }

        abstract void menu();
}
class Adopter extends User {
        private ArrayList<Pet> applications;

        public Adopter(String name, String password) {
        super(name, password);
        this.applications = new ArrayList<>();
        }
 public void apply(Pet pet, ArrayList<String> globalApplications) {
        applications.add(pet);
        globalApplications.add(getName() + " applied for " + pet.getName());
        }
 public void viewApplications() {
        if (applications.isEmpty()) {
        System.out.println("No applications submitted.");
        } else {
        System.out.println("Your applications:");
        for (Pet pet : applications) {
                pet.showInfo();
        }
```

```
        }
        }
 public void removeApplication(String petName) {
        applications.removeIf(pet -> pet.getName().equals(petName));
        }
 @Override
        void menu() {
        System.out.println("Adopter Menu: Browse, apply, or view applications.");
        }
}
class Staff extends User {
        public Staff(String name, String password) {
        super(name, password);
        }
 public void addPet(ArrayList<Pet> pets, String name, String type) {
        Pet pet = type.equals("dog") ? new Dog(name) : new Cat(name);
        pets.add(pet);
        System.out.println("Added " + name + " successfully!");
        }
 public void addPet(ArrayList<Pet> pets, String name, String type, String note) {
        Pet pet = type.equals("dog") ? new Dog(name) : new Cat(name);
        pets.add(pet);
        System.out.println("Added " + name + " with note: " + note);
        }
```

```
 @Override
        void menu() {
        System.out.println("Staff Menu: Add, approve, decline, or view
applications.");
        }
}
abstract class Pet {
        private String name;
        private boolean adopted;
 public Pet(String name) {
        this.name = name;
        this.adopted = false;
        }
 public String getName() {
        return name;
        }
 public boolean isAdopted() {
        return adopted;
        }
 public void setAdopted(boolean adopted) {
        this.adopted = adopted;
        }
abstract void showInfo();
}
```

```
class Dog extends Pet {
        public Dog(String name) {
        super(name);
        }
 @Override
        void showInfo() {
        System.out.println("Dog: " + getName() + ", " + (isAdopted() ? "Adopted" :
"Available"));
        }
}
class Cat extends Pet {
        public Cat(String name) {
        super(name);
        }
 @Override
        void showInfo() {
        System.out.println("Cat: " + getName() + ", " + (isAdopted() ? "Adopted" :
"Available"));
        }
}
public class PetAdoption {
        private static ArrayList<User> users = new ArrayList<>();
        private static ArrayList<Pet> pets = new ArrayList<>();
        private static ArrayList<String> applications = new ArrayList<>();
        private static Scanner scanner = new Scanner(System.in);
```

```
public static void main(String[] args) {
        users.add(new Staff("Admin", "Admin123"));
        users.add(new Adopter("Customer", "pet123"));
        System.out.println("Welcome to PetPal Adoption System!");
        while (true) {
        System.out.println("\n1. Login 2. Exit");
        System.out.print("Choose: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        if (choice == 1) {
                login();
        } else if (choice == 2) {
                System.out.println("Thank you for using PetPal!");
                break;
        }
        }
        }
private static void login() {
        System.out.print("Name (Admin/Customer): ");
        String name = scanner.nextLine();
        System.out.print("Password: ");
        String password = scanner.nextLine();
        for (User user : users) {
        if (user.getName().equals(name) && user.checkPassword(password)) {
```

```
            System.out.println("Welcome, " + name + "!");
              if (user instanceof Staff) {
              staffMenu((Staff) user);
              } else {
                  adopterMenu((Adopter) user);
              }
              return;
        }
        }
        System.out.println("Error: Invalid name or password.");
        }
 private static void staffMenu(Staff staff) {
        while (true) {
        System.out.println("\n1. Add Pet 2. Add Pet with Note 3. Approve Adoption 4.
Decline Application 5. Show Pets 6. View Applications 7. Logout");
        System.out.print("Choose: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        if (choice == 1) {
              System.out.print("Pet name: ");
              String name = scanner.nextLine();
              System.out.print("Type (dog/cat): ");
              String type = scanner.nextLine();
               staff.addPet(pets, name, type);
```

```
} else if (choice == 2) {
        System.out.print("Pet name: ");
        String name = scanner.nextLine();
        System.out.print("Type (dog/cat): ");
        String type = scanner.nextLine();
        System.out.print("Note: ");
        String note = scanner.nextLine();
        staff.addPet(pets, name, type, note);
} else if (choice == 3) {
        System.out.print("Pet name to approve: ");
        String name = scanner.nextLine();
        boolean found = false;
        for (Pet pet : pets) {
        if (pet.getName().equals(name)) {
            pet.setAdopted(true);
          System.out.println("Approved adoption for " + name + "!");
            found = true;
            break;
        }
        }
        if (!found) {
        System.out.println("Error: Pet not found.");
        }
} else if (choice == 4) {
```

```
            System.out.print("Pet name to decline: ");
            String name = scanner.nextLine();
            declineApplication(name);
        } else if (choice == 5) {
            showPets();
        } else if (choice == 6) {
            viewApplications();
        } else if (choice == 7) {
            break;
        }
        }
        }
 private static void adopterMenu(Adopter adopter) {
        while (true) {
        System.out.println("\n1. Browse Pets 2. Apply for Adoption 3. View
Applications 4. Logout");
        System.out.print("Choose: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        if (choice == 1) {
            showPets();
        } else if (choice == 2) {
            System.out.print("Pet name to apply: ");
            String name = scanner.nextLine();
```

```
        boolean found = false;
        for (Pet pet : pets) {
        if (pet.getName().equals(name)) {
                if (!pet.isAdopted()) {
                adopter.apply(pet, applications);
                System.out.println("Applied for " + name + "!");
                found = true;
                break;
                } else {
                System.out.println("Error: " + name + " is already adopted.");
                found = true;
                break;
                }
        }
        }
        if (!found) {
         System.out.println("Error: Pet not found.");
        }
} else if (choice == 3) {
        adopter.viewApplications();
} else if (choice == 4) {
        break;
}
}
```

```
        }
private static void showPets() {
        if (pets.isEmpty()) {
        System.out.println("No pets available.");
        } else {
        System.out.println("Available Pets:");
        for (Pet pet : pets) {
                pet.showInfo();
        }
        }
        }
private static void viewApplications() {
        if (applications.isEmpty()) {
        System.out.println("No adoption applications submitted.");
        } else {
        System.out.println("Adoption Applications:");
        for (String app : applications) {
              System.out.println(app);
        }
        }
        }
private static void declineApplication(String petName) {
        boolean found = false;
        for (String app : new ArrayList<>(applications)) {
```

```
        if (app.contains(" applied for " + petName)) {
                applications.remove(app);
                found = true;
        }
        }
        for (User user : users) {
        if (user instanceof Adopter) {
                ((Adopter) user).removeApplication(petName);
        }
        }
        if (found) {
        System.out.println("Declined application for " + petName + "!");
        } else {
        System.out.println("Error: No application found for " + petName + ".");
        }
        }
}
```

THE END