Gestion d'un Hôpital - Exercice Noté de Revision

Chaouki BAYOUDHI

29 décembre 2024

Introduction

Dans cet exercice modernisé, nous développons une application simplifiée de gestion d'un hôpital en utilisant les concepts avancés de la programmation orientée objet. Nous incluons :

- Les classes abstraites et l'héritage.
- Les exceptions personnalisées.
- L'utilisation de plusieurs interfaces, y compris des interfaces prédéfinies et personnalisées.
- L'utilisation de Lombok pour simplifier le code boilerplate.
- Une organisation claire et évolutive du code.

Instructions

Vous devez compléter le code fourni en respectant les numéros indiqués et en suivant les bonnes pratiques de programmation orientée objet.

Code Source

Classe abstraite `Person`

```java
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public abstract class Person {
    private int id;
    private String name;
    private String address;

    public abstract void displayInfo();
}
```

Classe `Patient`

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Patient extends Person implements Comparable<Patient> {
    private int socialSecurityNumber;
    private String ailment;

    @Override
    public void displayInfo() {
        // À compléter : Affichez les informations du patient.
```

```
    }

    @Override
    public int compareTo(Patient other) {
        // À compléter : Comparez deux patients par leur ID.
    }
}
```

Classe `Doctor`

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Doctor extends Person {
    private String specialty;
    private List<String> qualifications;

    @Override
    public void displayInfo() {
        // À compléter : Affichez les informations du médecin.
    }
}
```

Interfaces pour les Collections

```
public interface PatientCollection {
    void addPatient(Patient p); // À compléter
    void removePatient(Patient p); // À compléter
    Patient findPatientById(int id); // À compléter
    void displayAllPatients(); // À compléter
}

public interface DoctorPatientMap {
    void assignPatientToDoctor(Doctor d, Patient p) throws
        HospitalException; // À compléter
    List<Patient> getPatientsOfDoctor(Doctor d); // À compléter
    void displayAllAssignments(); // À compléter
}
```

Classe `Hospital`

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Hospital implements PatientCollection, DoctorPatientMap {
    private List<Patient> patients = new ArrayList<>();
    private Map<Doctor, Set<Patient>> doctorPatientMap = new HashMap
        <>();

    @Override
    public void addPatient(Patient p) {
        // À compléter
    }

    @Override
    public void removePatient(Patient p) {
        // À compléter
    }

```

```
18              @Override
19              public Patient findPatientById(int id) {
20                  // À compléter
21              }
22
23              @Override
24              public void displayAllPatients() {
25                  // À compléter
26              }
27
28              @Override
29              public void assignPatientToDoctor(Doctor d, Patient p) throws
                    HospitalException {
30                  // À compléter
31              }
32
33              @Override
34              public List<Patient> getPatientsOfDoctor(Doctor d) {
35                  // À compléter
36              }
37
38              @Override
39              public void displayAllAssignments() {
40                  // À compléter
41              }
42          }
```

Tâches Supplémentaires

1. Implémentez une classe `Specialist` qui hérite de `Doctor` avec un champ supplémentaire `expertiseArea`.
2. Ajoutez des méthodes statiques pour effectuer des analyses : `getDoctorsBySpecialty`, `getPatientsByAilment`.
3. Créez un programme principal pour tester toutes les fonctionnalités.
4. Ajoutez une méthode pour trier les patients par leur nom.
5. Implémentez une version de `PatientCollection` utilisant `Set` au lieu de `List`.
6. Testez le comportement des exceptions personnalisées dans différents scénarios.
7. Ajoutez des fonctionnalités de recherche avancée (par adresse, par spécialité, etc.).
8. Créez des rapports de statistiques : nombre de patients par médecin, maladies les plus fréquentes.
9. Implémentez une méthode pour exporter les données de l'hôpital au format JSON.