Projet Java - Système de réservation StarFleet

Contexte

L'année est 2371. La Fédération des Planètes Unies a besoin d'un système de réservation pour gérer les voyages à bord des vaisseaux de StarFleet. Vous êtes un ingénieur informatique récemment diplômé de l'Académie de StarFleet et on vous a confié la mission de développer ce système.

Objectif

Concevoir et implémenter un système de réservation en Java permettant de gérer les missions spatiales et les passagers à bord des vaisseaux de StarFleet.

Compétences mises en œuvre

- Programmation orientée objet (classes, objets)
- Héritage et polymorphisme
- Encapsulation
- Collections Java
- Entrées/sorties (lecture/écriture de fichiers)

- Design patterns simples

Structure du projet

```
StarFleetReservation/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── fr/
│   │   │   │   ├── starfleet/
│   │   │   │   │   ├── modele/
│   │   │   │   │   │   ├── personne/
│   │   │   │   │   │   │   ├── Personne.java
│   │   │   │   │   │   │   ├── Officier.java
│   │   │   │   │   │   │   └── Civil.java
│   │   │   │   │   │   ├── vaisseau/
│   │   │   │   │   │   │   └── Vaisseau.java
│   │   │   │   │   │   ├── mission/
│   │   │   │   │   │   │   └── Mission.java
│   │   │   │   │   │   └── reservation/
│   │   │   │   │   │       └── Reservation.java
│   │   │   │   │   ├── systeme/
│   │   │   │   │   │   └── SystemeReservation.java
│   │   │   │   │   ├── ui/
│   │   │   │   │   │   └── InterfaceConsole.java
│   │   │   │   │   ├── util/
│   │   │   │   │   │   ├── DateUtil.java
│   │   │   │   │   │   └── FileUtil.java
```

```
│   │   │   │   │   └── Main.java
├── data/
│   └── sauvegarde.dat
├── docs/
│   └── conception.md
├── README.md
```

Consignes

1. Structure des classes

Créez les classes suivantes :

Classe `Personne` (abstraite)

```java
public abstract class Personne {
    private String nom;
    private String prenom;
    private String identifiant;

    // Constructeur, getters, setters

    // Méthode abstraite
    public abstract String getDescription();
}
```

Classes dérivées de `Personne`

- `Officier` (avec rang et spécialité)
- `Civil` (avec planète d'origine et motif de voyage)

Classe `Vaisseau`

```java
public class Vaisseau {
    private String nom;
    private String immatriculation;
    private int capaciteMaximale;
    private List<Mission> missions;

    // Constructeur, getters, setters

    // Méthodes pour gérer les missions
    public void ajouterMission(Mission mission);
    public List<Mission> getMissions();
    // etc.
}
```

Classe `Mission`

```java
public class Mission {
    private String code;
    private String description;
    private Date dateDepart;
    private Date dateRetour;
    private String destination;
```

```
    private Vaisseau vaisseau;
    private List<Reservation> reservations;
    private int capaciteMaximale;

    // Constructeur, getters, setters

    // Méthodes de gestion des réservations
    public boolean ajouterReservation(Reservation reserv
    public boolean annulerReservation(String idReservati
    public int getNombrePlacesDisponibles();
    // etc.
}
```

Classe `Reservation`

```
public class Reservation {
    private String idReservation;
    private Personne passager;
    private Mission mission;
    private Date dateReservation;
    private boolean confirmee;

    // Constructeur, getters, setters

    // Méthodes
    public void confirmer();
    public void annuler();
    // etc.
}
```

Classe `SystemeReservation` (classe principale qui gère tout le système)

```
public class SystemeReservation {
    private List<Vaisseau> vaisseaux;
    private List<Personne> personnes;
    private List<Mission> missions;
    private List<Reservation> reservations;

    // Constructeur, getters, setters

    // Méthodes de gestion
    public void ajouterVaisseau(Vaisseau vaisseau);
    public void ajouterPersonne(Personne personne);
    public void creerMission(Mission mission);
    public Reservation effectuerReservation(String idPer
    public List<Mission> rechercherMissions(String desti
    public void sauvegarderDonnees(String fichier);
    public void chargerDonnees(String fichier);
    // etc.
}
```

2. Interface Utilisateur Console

Créez une classe `InterfaceConsole` qui permettra d'interagir avec le système via la console :

```
public class InterfaceConsole {
    private SystemeReservation systeme;
    private Scanner scanner;
```

```
    // Constructeur

    // Méthode principale pour démarrer l'interface
    public void demarrer();

    // Méthodes pour les différentes fonctionnalités
    private void afficherMenu();
    private void gererVaisseaux();
    private void gererPersonnes();
    private void gererMissions();
    private void gererReservations();
    private void sauvegarderDonnees();
    private void chargerDonnees();
    // etc.
}
```

3. Fonctionnalités à implémenter

1. **Gestion des vaisseaux**
   - Ajouter, modifier, supprimer, afficher les vaisseaux
2. **Gestion des personnes**
   - Ajouter, modifier, supprimer, afficher les membres d'équipage et les civils
3. **Gestion des missions**
   - Créer, modifier, annuler, afficher les missions
   - Associer une mission à un vaisseau
4. **Gestion des réservations**
   - Créer une réservation pour une personne sur une mission

- Confirmer/annuler une réservation
- Afficher les réservations d'une personne
- Afficher les réservations pour une mission

5. **Recherche**
   - Rechercher des missions par destination, dates, places disponibles
   - Rechercher des personnes par nom, identifiant
   - Rechercher des vaisseaux par nom, immatriculation
6. **Persistance des données**
   - Sauvegarder toutes les données dans un fichier
   - Charger les données depuis un fichier

Exemple de scénario de test

1. Créer plusieurs vaisseaux (USS Enterprise, USS Voyager)
2. Ajouter des officiers (Capitaine Picard, Commander Riker...) et des civils
3. Créer des missions vers différentes destinations (Vulcain, Terre, Risa...)
4. Effectuer des réservations pour différentes personnes
5. Confirmer/annuler certaines réservations
6. Afficher les listes de passagers pour chaque mission
7. Sauvegarder toutes les données dans un fichier
8. Quitter le programme
9. Relancer le programme et charger les données
10. Vérifier que tout fonctionne toujours

Implémentation de la sérialisation

Pour la persistance des données, vous pouvez utiliser la sérialisation Java :

```
// Exemple de sauvegarde
public void sauvegarderDonnees(String fichier) throws IO
    try (ObjectOutputStream oos = new ObjectOutputStream
            new FileOutputStream(fichier))) {
        oos.writeObject(this);
    }
}

// Exemple de chargement
public static SystemeReservation chargerDonnees(String f
    try (ObjectInputStream ois = new ObjectInputStream(
            new FileInputStream(fichier))) {
        return (SystemeReservation) ois.readObject();
    }
}
```

N'oubliez pas de faire implémenter `Serializable` à toutes vos classes qui doivent être sauvegardées.

Conseils

- Utilisez les interfaces et les classes abstraites pour tirer parti du polymorphisme
- Pensez à l'encapsulation pour protéger les données

- Utilisez les collections appropriées selon les besoins (ArrayList, HashMap, etc.)
- N'oubliez pas de gérer les exceptions pour les entrées/sorties
- Commentez votre code et suivez les bonnes pratiques de nommage

Livrable

Un projet Java complet contenant :

- Toutes les classes mentionnées ci-dessus
- Une documentation JavaDoc pour toutes les classes et méthodes principales
- Un rapport expliquant vos choix de conception
- Un fichier README expliquant comment lancer et utiliser votre application

Soumission du projet sur GitHub

1. Création d'un dépôt GitHub

1. Si vous n'avez pas encore de compte GitHub, créez-en un sur github.com
2. Connectez-vous à votre compte GitHub
3. Cliquez sur le bouton "+" en haut à droite, puis sur "New repository"

4. Nommez votre dépôt `starfleet-reservation-system`
5. Ajoutez une description (optionnel) : "Système de réservation pour vaisseaux StarFleet - Projet Java"
6. Choisissez l'option "Public"
7. Cochez "Add a README file"
8. Choisissez "Java" dans le menu déroulant "Add .gitignore"
9. Cliquez sur "Create repository"

2. Clonage du dépôt sur votre machine locale

```
git clone https://github.com/VOTRE_USERNAME/starfleet-re
cd starfleet-reservation-system
```

3. Structure de votre dépôt

Organisez votre dépôt selon la structure de projet recommandée ci-dessus.

4. Commits réguliers

Prenez l'habitude de faire des commits réguliers avec des messages clairs :

```
git add .
git commit -m "Description des modifications apportées"
```

Exemples de messages de commit :

- "Création de la classe abstraite Personne"
- "Implémentation de la gestion des réservations"
- "Correction du bug dans la méthode de recherche"

5. Pousser les modifications vers GitHub

```
git push origin main
```

6. Documentation du projet dans le README.md

Votre fichier README.md doit contenir au minimum :

- Titre et description du projet
- Nom et prénom de l'étudiant
- Instructions pour compiler et exécuter le programme
- Liste des fonctionnalités implémentées
- Captures d'écran de l'application en fonctionnement (si pertinent)
- Difficultés rencontrées et solutions apportées

7. Finalisation du projet

1. Assurez-vous que votre code est bien commenté et respecte les conventions de nommage Java
2. Vérifiez que tous vos fichiers sont bien poussés sur GitHub
3. Créez un tag pour marquer la version finale :

```
git tag -a v1.0 -m "Version finale du projet"
git push origin v1.0
```

8. Soumission du lien vers votre dépôt

Soumettez l'URL de votre dépôt GitHub (`https://github.com/VOTRE_USERNAME/starfleet-reservation-system`) selon les modalités indiquées.

Date de rendu

31 Mars 2025

Bon courage, cadet ! Que votre code vive longtemps et prospère !