Projet de Réservation de Billets de TGV

1. Objectifs du Projet

L'objectif principal est de créer une application console en C++ qui permet aux utilisateurs de réserver des billets pour un voyage en TGV. Le système doit gérer les réservations en prenant en compte différents paramètres, tels que le train, les horaires, les destinations, les places disponibles, les types de billets (première classe, deuxième classe), et les informations des passagers.

---

2. Classes Principales

a) Classe Train

- **Attributs** :
  - numeroTrain : identifiant unique du train.
  - villeDepart : ville de départ du train.
  - villeArrivee : ville de destination du train.
  - horaireDepart : heure de départ du train.
  - horaireArrivee : heure d'arrivée du train.
  - capacite : nombre total de places disponibles.
  - placesLibres : nombre de places encore disponibles.
- **Méthodes** :
  - verifierDisponibilite : vérifie si des places sont disponibles.
  - reserverPlace : réduit le nombre de placesLibres lorsqu'une réservation est effectuée.
  - annulerReservation : augmente le nombre de placesLibres lorsqu'une réservation est annulée.
  - afficherInfosTrain : affiche les informations du train (villes, horaires, disponibilité).

---

b) Classe Billet

- **Attributs** :
  - numeroBillet : identifiant unique du billet.
  - typeClasse : type de billet (première ou deuxième classe).
  - prix : prix du billet (calculé en fonction du type de classe et de la distance).
  - numeroTrain : identifiant du train pour lequel le billet est valide.
  - dateVoyage : date du voyage.
- **Méthodes** :
  - afficherDetailsBillet : affiche les informations du billet.
  - calculerPrix : calcule le prix du billet selon la classe et la distance entre les villes.

---

c) Classe Passager

- **Attributs** :
  - nom : nom du passager.
  - prenom : prénom du passager.
  - identifiant : identifiant unique du passager.
  - reservation : liste des billets réservés par le passager.
- **Méthodes** :
  - ajouterReservation : ajoute un billet à la liste de réservations.
  - annulerReservation : retire un billet de la liste de réservations.
  - afficherReservations : affiche les réservations actuelles du passager.

---

d) Classe Reservation

- **Attributs** :
  - numeroReservation : identifiant unique de la réservation.
  - passager : instance de la classe Passager associée à cette réservation.
  - billet : instance de la classe Billet pour la réservation.
- **Méthodes** :
  - confirmerReservation : finalise et confirme la réservation pour le passager.
  - annulerReservation : annule la réservation et met à jour la disponibilité des places du train.
  - afficherDetailsReservation : affiche les informations détaillées de la réservation (passager, billet, train, etc.).

---

3. Fonctionnalités de l'Application

a) Réservation de Billet

1. **Sélection du train** : L'utilisateur peut voir une liste des trains disponibles, avec les informations de chaque train (ville de départ, destination, horaires, disponibilité).
2. **Choix du billet** : Une fois le train sélectionné, l'utilisateur peut choisir le type de billet (première ou deuxième classe).
3. **Entrée des informations du passager** : L'utilisateur entre ses informations personnelles pour créer un compte de passager.
4. **Confirmation de la réservation** : Une réservation est générée, et le nombre de places disponibles est mis à jour. Un numéro de billet et de réservation sont attribués.

---

b) Annulation de Réservation

1. **Recherche de la réservation** : L'utilisateur peut retrouver une réservation en utilisant son numéro de réservation ou ses informations personnelles.
2. **Annulation** : La réservation est annulée, et le billet est supprimé de la liste des réservations du passager. Le nombre de places disponibles est également mis à jour.

---

c) Affichage des Réservations

1. **Liste des réservations du passager** : L'utilisateur peut voir toutes ses réservations actuelles, avec les détails de chaque billet.
2. **Détail des réservations** : Pour chaque réservation, l'utilisateur peut voir les détails, y compris les informations sur le train, le billet, et la classe de voyage.

---

4. Fonctionnalités Avancées

a) Affichage du Calendrier des Trains

Permet d'afficher un calendrier contenant les horaires de tous les trains disponibles sur une période donnée. Les utilisateurs pourront consulter les départs et arrivées pour des dates spécifiques.

- **Classe : Calendrier**
  - **Attributs** :
    - horaireTrain : un dictionnaire (ou map) où chaque clé représente une date et contient une liste des trains programmés ce jour-là.
  - **Méthodes** :
    - ajouterHoraire : ajoute un train et ses horaires au calendrier pour une date donnée.
    - afficherCalendrier : affiche tous les trains disponibles pour une période spécifique.

        - rechercherTrainsParDate : affiche les trains disponibles pour une date précise, filtrée par ville de départ et de destination.

---

b) Historique des Tickets des Passagers

Permet de garder un historique des tickets réservés par chaque passager, y compris les voyages passés.

- **Modifications dans la Classe Passager** :
    - Ajouter un attribut historiqueReservations qui stockera tous les tickets passés (terminés ou annulés) en plus des réservations actuelles.
- **Méthodes** :
    - ajouterHistorique : ajoute un billet dans l'historique lorsque le voyage est terminé ou que la réservation est annulée.
    - afficherHistorique : affiche tous les billets de l'historique d'un passager, avec les détails de chaque billet (date de voyage, ville de départ et d'arrivée, classe, prix, état – terminé ou annulé).
    - filtrerHistorique : permet de filtrer l'historique selon des critères comme la période, la ville de départ/destination, ou le type de billet.

---

5. Scénarios d'Utilisation

1. **Réserver un billet** :
    - Sélection du train, choix du billet, saisie des informations, confirmation.
2. **Annuler une réservation** :
    - Recherche et suppression d'une réservation existante.
3. **Consulter les réservations** :
    - Affichage de toutes les réservations en cours.
4. **Afficher le calendrier des trains** :
    - Consultation des horaires pour une période spécifique.
5. **Afficher l'historique des tickets** :
    - Affichage et filtrage des voyages passés et des annulations.

---

6. Gestion des Données et Persistance

Pour rendre l'application plus réaliste et fonctionnelle, il est nécessaire de gérer les données de manière persistante afin qu'elles soient disponibles même après la fermeture de l'application.

a) Stockage des Données

- Les données des trains, des passagers, des billets, et des réservations peuvent être stockées dans des fichiers texte ou au format JSON.
- Chaque fichier correspond à une entité :
    - trains.txt : contient les informations sur les trains.
    - reservations.txt : stocke les réservations effectuées.
    - passagers.txt : contient les informations des passagers.
    - historique.txt : contient l'historique des tickets de tous les passagers.

b) Lecture et Écriture dans les Fichiers

- **Lecture des données** : Lors du lancement de l'application, les fichiers sont lus pour initialiser les objets en mémoire.
- **Sauvegarde des données** : Après chaque action (réservation, annulation, mise à jour), les fichiers sont mis à jour pour garantir la persistance.

c) Méthodes associées :

- chargerDonnees : charge les données des fichiers dans les objets en mémoire.
- sauvegarderDonnees : écrit les données des objets dans les fichiers correspondants.

- exporterHistorique : permet d'exporter l'historique d'un passager au format texte ou CSV.

---

7. Gestion des Exceptions et des Erreurs

L'application doit inclure une gestion robuste des erreurs pour offrir une expérience utilisateur fluide.

a) Validation des Entrées Utilisateur

- Vérifier que les champs ne sont pas vides (nom, prénom, etc.).
- Valider les formats des dates, horaires, et numéros.

b) Gestion des Conflits

- **Réservation pleine** : Si un train n'a plus de places disponibles, un message d'erreur est affiché.
- **Annulation non trouvée** : Si l'utilisateur essaie d'annuler une réservation inexistante, le système doit en informer.

c) Gestion des Fichiers

- Si un fichier requis est manquant, le système crée un fichier vide par défaut.
- En cas d'erreur de lecture ou d'écriture, une exception est levée avec un message clair.

---

8. Scénarios d'Interaction Entre les Classes

1. **Réservation d'un billet** :
    - L'utilisateur sélectionne un train via la classe Calendrier.
    - La classe Billet génère les informations du ticket (prix, type de classe).
    - La classe Passager enregistre le billet dans sa liste de réservations.
    - La classe Reservation crée une réservation liée au billet et au passager.
2. **Annulation d'une réservation** :
    - L'utilisateur identifie sa réservation via la classe Reservation.
    - La classe Train libère une place.
    - La classe Passager retire le billet annulé et l'ajoute à l'historique.
3. **Affichage de l'historique** :
    - L'utilisateur demande à la classe Passager de filtrer et afficher son historique.
    - La classe Reservation récupère les informations des billets archivés.

---

9. Extensions Futures

L'application peut être étendue pour inclure :

1. **Système de Récompenses** : Gestion des points de fidélité pour les utilisateurs réguliers.

---

10. Livrable

Code c++ contenant les classes séparées en 2 fichiers .cpp et .h
Rapport PDF contenant une description des sénarios du projet