POO _ Travail Pratique (Partie 2)

ESI-RUN

Durée 2h- documentation autorisée / Utilisation d'internet interdite

---

Important

- L'objectif de cette séance est d'implémenter la partie de l'application qui sert à gérer les titres de transport.
- **Consignes à respecter avant de commencer**
  - Assurez vous d'avoir téléchargé les fichiers .java des classes `TitreTransport`, `Personne`, `Usager` et `Employe`
  - Téléchargez le fichier TestTransport. java comportant la méthode main de votre projet. Assurez-vous de ne pas modifier les noms des classes et des méthodes, leurs signatures et types de retour.
  - Lisez attentivement l'énoncé ainsi que le code de la méthode main.
- **Consignes à respecter lors de la programmation:**
  - Toutes les classes doivent être publiques et appartenir au package "core"
  - La classe TestTransport comportant la méthode main doit appartenir au package "test"
- **Consignes à respecter avant le dépôt de la solution**:
  - Le code source (.java) doit être placé dans un répertoire nommé src (contenant tous les packages, comme indiqué sur l'image fournie dans le mail de consignes).
  - Les fichiers compilés (.class) dans un répertoire bin (comme indiqué sur l'image fournie dans le mail de consignes).
  - **Compressez** le répertoire **src** et renommer l'archive en : numéroBinome**src**.zip (ex. : 1**src**.zip).
  - Faites une **copie du répertoire bin**, renommez cette copie avec le numéro du binôme (ex. : 1), puis compressez ce répertoire (ex. : 1.zip).
  - Remettez les deux fichiers .zip sur le lien Moodle dédié à ce livrable, ainsi que sur le lien GoogleForm dédié à chaque groupe, comme indiqué dans le mail de l'énoncé.

Enoncé

Pour se déplacer dans le réseau de transport, un usager doit acquérir un titre de transport. Un titre de transport est caractérisé par son identifiant unique, sa date et heure d'achat et son prix. Avant d'emprunter un moyen de transport, l'usager doit valider son titre via le lecteur magnétique présent dans la station. Si le titre de transport n'est pas valide, une erreur est signalée et l'accès est refusé.

Notre système distingue deux types de titres de transport.

1. Le ticket : il coûte 50 DA et valable pour un seul voyage le jour de son acquisition (achat)
2. Carte de navigation personnelle: Elle est valable pour une durée d'une année et appartient à un usager unique. Plusieurs types de cartes personnelles sont disponibles dont le prix est calculé en appliquant une réduction sur le prix initial de 5000 DA comme suit.
    - Carte junior offrant une réduction de 30 % pour les usagers de moins de 25 ans
    - Carte Senior offrant une réduction de 25 % pour les usagers de plus de 65 ans
    - Carte solidarité offrant une réduction de 50 % pour les Personnes à mobilité réduite.
    - Carte partenaire offrant une réduction de 40% pour les employés de l'entreprise.

Toute personne désirant acquérir une carte de navigation personnelle doit donner son nom, son prénom, sa date de naissance et mentionner si elle souffre d'un handicap. Si cette personne n'est pas employée au sein de la compagnie un numéro séquentiel lui est automatiquement attribué comme identifiant, sinon, elle doit fournir son numéro de matricule et indiquer sa fonction (Agent de station ou conducteur). Dans le cas où la personne n'a droit à aucune réduction, la carte n'est pas créée et une erreur est signalée. Si, au contraire, elle a droit à plusieurs réductions, elle bénéficie de la plus avantageuse. Par exemple, un employé dans l'entreprise souffrant d'un handicap aura une carte "solidarité".

Travail demandé: En vous basant sur cet énoncé et le code de la méthode main donné ci-dessous, ecrivez un programme permettant de :

- Créer des usagers, des employés, des tickets et des cartes de navigation personnelles.
- Vérifier la validité d'un titre de transport, la possibilité de bénéficier d'une carte de navigation personnelle pour un usager donné et choisir le type adéquat.

Annexe 1 — Exemple d'un scénario de test

```
package test;          //affecter la classe au package test
import core.*;         //importer le package comportant les classes du projet
import java.time.*;    //package pour manipuler les dates, heures, etc.

public class TestTransport {

  public static void main(String[] args) {

   TitreTransport ticket, cartePersoUsager,cartePersoEmploye;
   Personne usager, employe,usagerRExcept;

   System.out.println(" ---------------------- Tests sur les tickets----------------------");

   ticket = new Ticket();
   // 1.Test de ticket invalide
   try {
       System.out.println("Ticket créé le : " + ticket.getDateAchat ());
       System.out.println("Numéro du ticket  :  " + ticket.getId () );
       System.out.println("Prix du ticket : " + ticket.getPrix() + " DA");
       System.out.println("Test de validité le "+ LocalDate.of(2025,4,22)+ ": " +
ticket.estValide(LocalDate.of(2025,4,22)) );
       System.out.println("Test de validité le "+ LocalDate.of(2025,4,30)) ;
       System.out.println(ticket.estValide(LocalDate.of(2025,4,30))) ;

    } catch (TitreNonValideException e) {System.out.println("Erreur: " + e.getMessage());}

   System.out.println(" \n\n\n---------------------- Tests sur les cartes
personnelles----------------------");

   //2. Test des cartes personnelles
      usager = new Usager("Ali", "BenMohamed", LocalDate.of(2010, 5, 12), false);
      employe = new Employe("Ahmed", "Tahar", LocalDate.of(1980, 3, 25), false, "A123",
TypeFonction.AGENT);
      usagerRExcept = new Usager("Amina", "Hamidi", LocalDate.of(1980, 5, 12), false);

       //2.1. carte usager reduction possible: test de type et test de validité
try{
      cartePersoUsager = new CartePersonnelle(usager);
      System.out.println("Carte personnelle créée le " + cartePersoUsager.getDateAchat());
      System.out.println("Numéro de la carté: "+ cartePersoUsager.getId());
      System.out.println("Proprietaire de la carte : "+ usager.getNom()+ " " +
usager.getPrenom());
      System.out.println("Type de la carte " + ((CartePersonnelle)cartePersoUsager).getType() );
      System.out.println("Prix de la carte : " + cartePersoUsager.getPrix() + " DA");
      System.out.println("Test de validité le "+ LocalDate.of(2025,5,22) + " : " +
cartePersoUsager.estValide((LocalDate.of(2025,5,22))));
      System.out.println("Test de validité le "+ LocalDate.of(2026,5,22)) ;
      cartePersoUsager.estValide((LocalDate.of(2026,5,22)));

        }
  catch (TitreNonValideException e) {
       System.out.println("Erreur: " + e.getMessage()); }
  catch (ReductionImpossibleException e) {
       System.out.println("Erreur: " + e.getMessage()); }


        //2.2. Ceation de carte personnelle impossible (la personne n'a droit à aucune
réduction)
try{
       cartePersoUsager = new CartePersonnelle(usagerRExcept);
   }

  catch (ReductionImpossibleException e) {
       System.out.println("\nCréation de carte personnelle refusée pour "+
usagerRExcept.getNom()+ " " + usagerRExcept.getPrenom()+ e.getMessage());
       }
    }
}
```

Annexe 2 — Résultat de l'exécution du scénario de test

```
---------- Tests sur les tickets----------
Ticket créé le : 2025-04-22
Numéro du ticket  :  1
Prix du ticket : 50.0 DA
Test de validité le 2025-04-22: true
Test de validité le 2025-04-30
Erreur: Ticket numéro 1 expiré - valable uniquement le : 2025-04-22



---------- Tests sur les cartes personnelles----------
Carte personnelle créée le 2025-04-22
Numéro de la carté: 2
Proprietaire de la carte : Ali BenMohamed
Type de la carte JUNIOR
Prix de la carte : 3500.0 DA
Test de validité le 2025-05-22 : true
Test de validité le 2026-05-22
Erreur: Carte personnelle numéro  2 invalide

Création de carte personnelle refusée pour Amina Hamidi. Vous n'avez droit à
aucune réduction.
```

Annexe 3 — Aide à l'utilisation de java.time

Méthode	Fonction
`LocalDate.now()`	Retourne la date courante (sans l'heure).
`LocalDateTime.now()`	Retourne la date et l'heure courantes.
`LocalDate.isEqual(LocalDate)`	Vérifie si deux dates sont identiques.
`LocalDate.plusMonths(long)`	Ajoute un nombre de mois à une date.
`Period.between(date1, date2).getYears()`	Calcule la différence en années entre deux dates.
`date1.toLocalDate().equals(date2.toLocalDate())`	Vérifie si deux dates sont le même jour (sans l'heure).
`LocalDateTime.now().isAfter(dateExpiration)`	Vérifie si une date d'expiration est dépassée.
`LocalDate.of(année, mois, jour)`	Crée une date précise sans l'heure.