Nom :
Prénom :
Numéro d'étudiant :

Tous documents sur support papier autorisés. L'ensemble des réponses est à donner sur les feuilles d'énoncé.

L'aéroport imaginaire de Eldamar désire informatiser le suivi des voyageurs qui entrent dans le pays par le point de contrôle de police. Une énumération et trois premières classes vous sont données ci-dessous. Elles représentent respectivement : la décision prise lors du contrôle (admission, non admission), les passeports, les visas et un programme montrant la création de passeports et de visas. Vous aurez à compléter ce programme dans la suite de ce sujet.

```
public enum PoliceControlResult {authorizedEntry, noAdmission}

public class Passport {
   private String number="zzz"; // numéro de passeport
   private int deliveryDate=0;  // date de délivrance
   private int expiryDate=0;    // date d'expiration
   public Passport() {}
   public Passport(String number, int deliveryDate, int expiryDate) {
      this.number = number;
      this.deliveryDate = deliveryDate; this.expiryDate = expiryDate;
   }
   public String getNumber() {return number;}
   public void setNumber(String number) {this.number = number;}
   public int getDeliveryDate() {return deliveryDate;}
   public void setDeliveryDate(int deliveryDate) {this.deliveryDate = deliveryDate;}
   public int getExpiryDate() {return expiryDate;}
   public void setExpiryDate(int expiryDate) {this.expiryDate = expiryDate;}
}

public class Visa {
   private String visaNumber="zzz";  // numéro du visa
   private int entryDate=0;         // date après laquelle on peut entrer dans le pays
   private int exitDate=0;         // date à laquelle il faut être sorti du pays
   private String country="zzz";  // pays pour lequel le visa est valable
   public Visa() {}
   public Visa(String visaNumber, int entryDate, int exitDate, String country) {
      this.visaNumber = visaNumber; this.entryDate = entryDate;
      this.exitDate = exitDate; this.country = country;
   }
   public String getVisaNumber() {return visaNumber;}
   public void setVisaNumber(String visaNumber) {this.visaNumber = visaNumber;}
   public int getEntryDate() {return entryDate;}
   public void setEntryDate(int entryDate) {this.entryDate = entryDate;}
   public int getExitDate() {return exitDate;}
   public void setExitDate(int exitDate) {this.exitDate = exitDate;}
   public String getCountry() {return country;}
   public void setCountry(String country) {this.country = country;}
}
```

```
public class ProgrammeAeroport {
   public static void main(String[] args) {
      Passport pGil = new Passport("33TH45",20100102,20200101);
      Passport pEld = new Passport("36GD48",20121002,20220901);
      Visa vGil = new Visa("77",20140101,20180101,"Eldamar");
      Visa vEld = new Visa("88",20160101,20161231,"Himlad");
      .....}
}
```

Question 1. Ecrire une interface.
On souhaite représenter un voyageur arrivant à Eldamar par : (1) une méthode `getNom` permettant de connaître le nom du voyageur sous forme d'une chaîne de caractères, (2) une méthode `getPassport` permettant de connaître le passeport du voyageur, (3) une méthode `verify` qui prend comme paramètre un entier représentant une date et retourne une décision (valeur de l'énumération décrite plus haut).
Ecrivez une interface `IVoyageur` représentant les objets disposant de ces trois méthodes.
Les dates sont représentées sous la forme AAAAMMDD, par exemple l'entier 20160518 représente le 18 mai 2016.

Question 2. Implémenter une interface.
Ecrivez une classe abstraite `AbsVoyageur` qui implémente l'interface `IVoyageur`. Un voyageur a un nom (chaîne de caractères) et un passeport. La méthode `verify(int d)` retourne `authorizedEntry` si la date `d` est comprise entre la date de délivrance et la date d'expiration du passeport, sinon elle retourne `noAdmission`. N'écrivez que l'entête de la classe, les attributs, les méthodes nécessaires pour implémenter l'interface et un seul constructeur qui initialise les deux attributs.

Question 3. Spécialiser une classe abstraite.
Ecrivez une classe concrète `VoyageurBrethil` qui hérite de `AbsVoyageur`. Un voyageur en provenance de Brethil peut ou non disposer d'une invitation (booléen). La méthode `verify(int d)` retourne `authorizedEntry` si la date `d` est comprise entre la date de délivrance et la date d'expiration du passeport (comme dans la super-classe) et si de plus le voyageur dispose d'une invitation. Sinon elle retourne `noAdmission`. N'écrivez que l'entête de la classe, l'attribut, le constructeur qui initialise les attributs et la méthode `verify`.

Question 4. Ecrire une structure de données. Ecrivez une classe `PoliceCheckQue` pour représenter les files d'attente de voyageurs d'un point de contrôle :

1. les voyageurs sont stockés dans une `ArrayList`
2. un attribut `date` mémorise la date à laquelle la file d'attente est constituée
3. une méthode `boolean isEmpty()` retourne vrai si la file d'attente est vide, faux sinon
4. une méthode `void enter(IVoyageur v)` permet d'ajouter un voyageur à la fin de la file d'attente
5. une méthode `String control()` permet (1) d'enlever le premier voyageur de la file (2) de retourner une chaîne de caractères contenant le nom du voyageur et la décision prise par le point de contrôle (`authorizedEntry` ou `noAdmission`).

Question 5. Ecrire une assertion. Réécrivez la méthode `void enter(IVoyageur v)` en lui ajoutant, à la fin, une assertion vérifiant que `v` est bien dans la `ArrayList` de la file d'attente.

Question 6. Ecrire une exception. La pré-condition de la méthode `String control()` est que la file d'attente ne soit pas vide. Ecrivez une classe d'exception pour représenter cette erreur, puis modifiez la méthode `control` pour qu'elle signale cette exception lorsque cette pré-condition n'est pas remplie.

Question 7. Rendre une classe générique

Transformer la classe `PoliceCheckQue` en classe générique paramétrée par un type de voyageur. Vous devez observer quelle(s) méthode(s) est(sont) appelée(s) sur les voyageurs d'une file d'attente et en tirer les conséquences sur le contrat / la spécification que ces objets doivent respecter. Cette classe générique permettra de constituer des files d'attente homogènes avec des voyageurs en provenance d'un même endroit.