# Examen Ecrit - session 2

Tous documents sur support papier autorisés. Durée 2h.
L'ensemble des réponses sera à donner sur les feuilles d'énoncé. Ne pas dégrapher les feuilles.

**Question 1. Ecrire une interface `IActiviteCulturelle`.**

On souhaite représenter les activités culturelles proposées par une association locale dans différents secteurs (théâtre, cinéma, musique, etc.) qui disposent : (1) d'une méthode `getIntitulé` permettant de connaître leur nom sous forme d'une chaîne de caractères, (2) d'une méthode `getAbonnement` permettant de connaître le montant annuel de l'abonnement à l'activité, (3) d'une méthode `estSubventionnee` retournant vrai si l'activité est subventionnée par la mairie (faux sinon). Ecrivez le code Java de l'interface représentant les objets disposant de ces méthodes.

**Question 2. Compléter l'interface `IActiviteCulturelle`.**
On rappelle que l'API Java fournit une interface prédéfinie pour représenter les objets comparables : `public Interface Comparable<T> {int compareTo(T o)}`. Comme l'indique la documentation, dans une implémentation, la méthode `compareTo` doit retourner *a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.* Compléter le code de l'interface `IActiviteCulturelle` (indiquez les modifications sans tout recopier) de manière à ce qu'elle spécialise l'interface `Comparable`, pour représenter le fait que les activités culturelles sont comparables entre elles d'après leur intitulé. Ajustez le paramètre `T` au problème et implémentez la méthode `compareTo` avec une méthode `default` dans l'interface `IActiviteCulturelle`.

**Question 3. Implémenter une interface**.
(1) Ecrivez une classe `ActiviteCulturelle` qui implémente l'interface `IActiviteCulturelle`. N'écrivez que les attributs et les méthodes nécessaires pour implémenter l'interface. (2) Puis écrivez l'en-tête d'une classe `ActiviteThéatre` qui est sous-classe de `ActiviteCulturelle` (uniquement l'en-tête). Pour la suite, leurs constructeurs et accesseurs sont tous supposés exister sous leur forme classique (nom, types).

**Question 4. Utiliser une classe de l'API et écrire une structure de données**
Nous introduisons la classe générique `TreeSet<T>` de l'API standard de Java. Cette classe permet de représenter des ensembles à l'aide d'une structure interne arborescente. Un ensemble est une collection d'objets sans doublons (pour savoir si des objets sont identiques, la méthode `compareTo` de la classe des objets stockés dans l'ensemble est appelée). `TreeSet` dispose entre autres des méthodes suivantes :

- un constructeur sans paramètre pour créer un ensemble vide
- s.size() retourne le nombre d'éléments
- s.add(e) — adds e to s if e is not already present.
- s.contains(e) — returns true if this set contains e.
- s.remove(e) — removes the specified element e from s if it is present. returns true if s contained e.
- s1.containsAll(s2) — returns true if s2 is a subset of s1. (s2 is a subset of s1 if set s1 contains all of the elements in s2.)
- s1.addAll(s2) — transforms s1 into the union of s1 and s2. (The union of two sets is the set containing all of the elements contained in either set.)

Utilisez la classe `TreeSet<T>` pour réaliser une classe générique `PassCulture`. Un *Pass Culture* appartient à une personne (dont le nom sera stocké comme un attribut de type chaîne de caractères). Il contient l'ensemble des activités culturelles auxquelles participe la personne.

On désire représenter des Pass Culture contenant un ensemble d'activités hétérogènes (instances de plusieurs sous-classes d'`ActiviteCulturelle` comme `ActiviteTheatre`, `ActiviteMusique`, etc.) ou homogènes (instances d'une même sous-classe d'`ActiviteCulturelle`, par exemple instances de la sous-classe `ActiviteTheatre` comme {spectacles de théâtre, atelier d'improvisation théatre, *etc.*}. Quelle que soit la situation, un Pass Culture ne contient que des activités culturelles.

Dans cette question, écrivez (1) l'entête de la classe générique `PassCulture` (2) ses attributs en les initialisant.

**Question 5. Ecrire une méthode d'ajout**
(1) Ecrivez dans la classe `PassCulture` une méthode permettant d'ajouter à un Pass Culture une activité culturelle.
(2)) Pourquoi n'est-il pas nécessaire de vérifier dans votre méthode la présence d'une activité avant de l'ajouter au `TreeSet` ?

**Question 6. Ecrire une méthode de retrait et une assertion**
Ecrivez dans la classe `PassCulture` une méthode permettant de retirer à un Pass Culture une activité culturelle et retournant vrai si l'élément était effectivement présent (avant le retrait). Dans la méthode, prévoyez une assertion vérifiant la manière dont la taille (nombre d'éléments) de l'ensemble d'activités culturelles a évolué (analysez tout d'abord de quelle manière il a pu évoluer).

**Question 7. Comprendre les streams et ajouter une méthode manipulant la structure**
En supposant que l'ensemble des activités de la carte se nomme `listeActivites`. On peut écrire la méthode suivante basée sur les `streams`.

```
public double calcul(){
    return listeActivites
      .stream()
      .filter(act -> act.getAbonnement()>10 && act.estSubventionnee())
      .mapToDouble(act -> act.getAbonnement())
      .average()
      .getAsDouble();
}
```

Expliquez ce qu'elle fait et réécrivez cette méthode avec une itération classique (`for` ou `while`).

**Question 8. Ecrire une exception**

Nous nous intéressons à présent à écrire une méthode statique `absorbe` prenant en paramètre deux Pass Culture et ajoutant au premier toutes les activités du second. Cette méthode ne doit pas modifier le second Pass Culture passé en paramètre. Sa pré-condition indique que l'absorption ne peut être effectuée que si le premier Pass Culture ne contient pas déjà toutes les activités du second, sinon une exception est signalée. (1) Ecrivez une classe d'exception pour représenter cette erreur, puis (2) écrivez la méthode `absorbe` pour qu'elle déclare et signale cette exception lorsque la pré-condition n'est pas remplie. Consultez les méthodes de `TreeSet` présentées à la question 4 et choisissez celles vous permettant d'écrire rapidement ce code.

**Question 9. Ecrire un main**

Ecrivez un main : (1) Créez trois activités théâtre (spectacle, atelier_impro, atelier_diction) (2) Créez deux Pass Culture (3) Appelez la méthode `absorbe` d'une manière normale et d'une manière qui provoque le signalement d'exception (4) Capturez la ou les exceptions qui risquent de se produire dans le programme.