Le barème est indicatif, il peut varier légèrement. Il est fortement conseillé de lire le sujet en entier avant de commencer à coder. Les 3 derniers exercices sont indépendants les uns des autres.

Vous rendrez le répertoire du projet contenant les différentes classes que l'on vous demande d'écrire. Pour chaque question qui n'est pas du code, vous devez répondre en commentaire au dessus de la méthode correspondante.

Précisions : Vous n'avez pas le droit de mettre dans le code des méthodes d'instance publiques qui ne sont pas demandées dans le sujet. Vous devez faire en sorte que votre code ne puisse pas provoquer d'erreur "non voulues" s'il est utilisé correctement. Vous n'avez pas le droit d'utiliser des concepts qui n'ont pas été vus en cours.

Dans cet examen, nous allons modéliser des spectacles musicaux et en particulier des différentes personnes qui y participent.

Exercice 1. Les musiciens (∼ 3 points)

On souhaite représenter des musiciens (classe `Musician`). Un musicien est caractérisé par l'instrument dont il joue (une chaîne de caractères `instrument`), ainsi que l'ensemble des styles qu'il peut jouer, sous la forme d'une chaîne de caractères (`styles`), telle que les différents styles sont séparés par le caractère `'|'`. Il a enfin un tarif de base (donné par un entier).

Écrire la classe `Musician` dans le package `fr.dut.music` afin de pouvoir :

1. Créer un musicien avec ses différentes caractéristiques et un tarif strictement positif. Que doit-on faire si le tarif est négatif ? Pourquoi ?
2. Afficher un musicien de la façon suivante (le tarif ne figure pas dans l'affichage) :

```
Musician bob = new Musician("Reggae|Pop", "Guitar", 42);
System.out.println(bob);
```

```
Guitar player (Reggae|Pop)
```

3. Tester l'égalité de musiciens (le tarif n'est pas pris en compte pour le test) en respectant le "contrat objet". Quelle méthode doit-on obligatoirement ajouter après celle-ci et pourquoi ? Ajouter cette méthode dans le code.
4. Ajouter la méthode `cost` qui permet d'obtenir le coût d'un musicien qui correspond à son tarif de base multiplié par la longueur de la chaîne de caractères `styles`. Par exemple :

```
System.out.println(bob.cost());
```

```
420
```

5. Écrire les tests dans une classe `Test`, dans le package `fr.dut.music.test`.

Exercice 2. Le spectacle (∼ 4 points)

On veut maintenant représenter un spectacle (classe `Show`) avec les artistes qui y participent.

Écrire le code de la classe `Show` dans le package `fr.dut.music.show` :

1. On doit pouvoir construire un spectacle avec un style (chaîne de caractères) donné. Ajouter une méthode permettant de connaître le style du spectacle.
2. Fournir une méthode `addArtist` qui permet d'ajouter un musicien dans le spectable et qui renvoie un booléen. On peut ajouter un musicien seulement si le style du spectacle figure dans la liste de styles de ce musicien. Dans le cas contraire, la méthode doit lancer une exception. Enfin, on ne peut pas ajouter un musicien qui participe déjà au spectacle : dans cas, la méthode renvoie `false`, sinon elle renvoie `true`.

   Pourquoi est-il préférable d'empêcher l'ajout d'un objet `null` dans une collection ?

3. Ajouter une méthode permettant de connaître les nombre de musiciens qui participent au spectacle.
4. Ajouter une méthode `costListing` qui renvoie une chaîne de caractères qui donne la liste des musiciens participant au spectacle, avec leur coût, ainsi que le coût total du spectacle (la somme des coûts des musiciens), formatée comme dans l'exemple ci-dessous :

```
Show show = new Show("Pop");

Musician bob = new Musician("Reggae|Pop", "Guitar", 42);
show.addArtist(bob);

show.addArtist(new Musician("Reggae|Pop", "Guitar", 57)); // false, déjà là.
show.addArtist(new Musician("Pop|Jazz|Rock", "Guitar", 500));

// cette ligne devrait lever une exception car ce musicien ne fait pas de Pop
// show.addArtist(new Musician("Punk", "Bass", 100));

show.addArtist(new Musician("Pop", "Violin", 200));

System.out.println(show.costListing());
```

```
--------------------------
420 Guitar player (Reggae|Pop)
600 Violin player (Pop)
6500  Guitar player (Pop|Jazz|Rock)
--------------------------
7520
```

Exercice 3. Les chanteurs (∼ 4 points)

On souhaite ajouter des chanteurs (classe `Singer`) dans le spectacle. Un chanteur est caractérisé par son nom de scène, l'ensemble des styles qu'il peut chanter (même format que pour les musiciens) et l'étendue de sa renommée, représentée par un caractère ('I' pour internationale, 'N' pour nationale et 'L' pour locale ; toute autre lettre est invalide).

Écrire le code de la classe `Singer` dans le package `fr.dut.music` et modifier le reste du code afin de pouvoir effectuer les actions suivantes :

1. Créer un chanteur avec ses différentes caractéristiques.
2. Afficher un chanteur de la façon suivante (le nom est entre 3 étoiles pour les chanteurs de renommée internationale, 2 étoiles pour une renommée nationale et 1 étoile sinon) :

```
Singer joe = new Singer("Rock|Pop", "Johnny", 'N');
System.out.println(joe);
```

```
**Johnny** (Rock|Pop)
```

3. Renvoyer le coût d'un chanteur qui correspond à :
   — l'indice de renommée (3 pour internationale, 2 pour nationale et 1 sinon) au carré,
   — multiplié par 10 fois le code ascii de la première lettre de son nom,
   — multiplié par la longueur de la chaîne de caractères `styles`.

   Par exemple (le code ascii de 'J' est 74) :

```
System.out.println(joe.cost());
```

```
23680
```

4. Permettre d'ajouter un chanteur dans un spectacle (sans ajouter de méthode supplémentaire dans `Show`). Les conditions d'ajout sont les mêmes que pour les musiciens. Le listing de coût doit faire apparaître tous les participants.
5. **Si ce n'est pas déjà fait, faire en sorte de ne pas dupliquer de code.**

Exercice 4. Les techniciens (∼ 3 points)

Un bon spectacle ne peut pas fonctionner sans techniciens (classe `Technician`). On veut donc pouvoir les ajouter au spectacle. Un technicien est caractérisé par son domaine de compétences représenté par un type `enum Field` qui peut prendre une des 4 valeurs suivantes : LIGHT, SOUND, COSTUME, DECOR.

Écrire le code de la classe `Technician` dans le package `fr.dut.music` et modifier le reste du code afin de pouvoir effectuer les actions suivantes :

1. Créer un technicien et l'afficher de la façon suivante :

```
System.out.println(new Technician(Technician.Field.SOUND));
```

```
SOUND tech.
```

2. Renvoyer le coût du technicien. Il s'agit d'un coût fixe : 50 000 pour les techniciens DECOR, 11 000 pour les techniciens COSTUME et 2 000 pour les autres.
3. Permettre d'ajouter un technicien au spectacle. Il n'y a pas de restriction sur leur ajout.
4. Les techniciens doivent apparaître dans les listing de coût. De plus, pour simplifier la vie du comptable, le listing doit désormais faire apparaître les participants **en ordre croissant de coût**. Par exemple :

```
Show show = new Show("Pop");
show.addTechnician(new Technician(Technician.Field.SOUND));
show.addArtist(new Musician("Reggae|Pop", "Guitar", 42));
show.addArtist(new Musician("Pop|Jazz|Rock", "Guitar", 500));
show.addTechnician(new Technician(Technician.Field.DECOR));
show.addTechnician(new Technician(Technician.Field.COSTUME));
show.addArtist(new Musician("Pop", "Violin", 200));
show.addArtist(new Musician("Pop|Jazz|Blues|Funk", "Ukulele", 1000));
show.addArtist(new Singer("Opera|Pop", "Maria Callas", 'I'));
show.addArtist(new Singer("Rock|Pop", "Johnny", 'N'));
show.addTechnician(new Technician(Technician.Field.SOUND));
System.out.println(show.costListing());
```

```
---------------------------
420 Guitar player (Reggae|Pop)
600 Violin player (Pop)
2000  SOUND tech.
2000  SOUND tech.
6500  Guitar player (Pop|Jazz|Rock)
11000 COSTUME tech.
19000 Ukulele player (Pop|Jazz|Blues|Funk)
23680 **Johnny** (Rock|Pop)
50000 DECOR tech.
62370 ***Maria Callas*** (Opera|Pop)
---------------------------
177570
```

Exercice 5. Fichier des artistes(∼ 4 points)
On dispose d'un fichier d'artistes au format suivant :

- un artiste par ligne
- sur chaque ligne :
  - la liste des styles sous la forme d'une chaîne de caractères telle que les différents styles sont séparés par le caractère `'|'` ;
  - l'instrument si c'est un musicien ou le nom de scène sinon ;
  - le tarif si c'est un musicien ou la renommée (un seul caractère) sinon.
- ces trois champs sont séparés par des points-virgule.

Par exemple :

```
Reggae|Pop;Guitar;42
Pop|Jazz|Rock;Guitar;500
Opera|Pop;Maria Callas;I
Pop;Violin;200
Pop|Jazz|Blues|Funk;Ukulele;1000
Rock|Pop;Johnny;N
```

Dans la classe `Show`, écrire une méthode `makeShowFromFile` qui prend en paramètre le chemin d'un nom de fichier, un style et un coût maximum. Cette méthode doit renvoyer un nouveau spectacle créé avec le style donné en paramètre, et en incluant des artistes dans l'ordre du fichier, en respectant les contraintes d'ajout, jusqu'à ce qu'il n'y ait plus d'artiste ou que l'ajout d'un artiste ne soit pas possible car il ferait dépasser le coût maximum donné en paramètre. Attention, ce fichier peut être très gros, vous ne devez pas le lire en entier si ce n'est pas nécessaire. Si le fichier est mal formé, la méthode doit lancer une `IllegalArgumentException`.

- Quelle est la particularité de cette méthode (dans sa déclaration) et pourquoi doit-elle être déclarée ainsi ?
- Comment fait-on pour créer un chemin à partir d'un nom de fichier ?
- Quel est l'intérêt d'utiliser un flux bufferisé pour la lecture de fichier ?

Exercice 6. Les chansons du spectacle (∼ 4 points)
L'organisateur du spectacle souhaite pouvoir proposer un programme avec des chansons qui attireront le public. Il veut mettre en place un système de vote permettant à n'importe qui d'indiquer sa chanson préférée, afin de sélectionner un ensemble des chansons les plus populaires.

1. Dans la classe `Show`, écrire une méthode `vote` qui prend en paramètre un titre de chanson (une chaîne de caractères) et permet de comptabiliser un vote pour cette chanson.
2. Ajouter une méthode `voteResult` qui affiche ligne par ligne chaque chanson qui ont reçu au moins un vote avec la méthode `vote`, avec le nombre de fois où l'on a voté pour elle. Les chansons devront être affichées en ordre alphabétique.
3. On veut empêcher une personne de voter plusieurs fois pour la même chanson. Pour cela, modifier la méthode vote pour qu'elle prenne comme deuxième paramètre le nom de la personne qui vote, puis modifier le code pour qu'il vérifie qu'une personne ne peut voter qu'une seule fois pour la même chanson (si une personne triche, ses votes supplémentaires ne sont simplement pas comptabilisés).