

**Exercice I (T9 et classes)**

On a vu dans le TD précédent comment implémenter des méthodes qui permettent de stocker des chaînes dans un dictionnaire dont les clés sont leur code T9. Certains éléments vus à ce moment là peuvent être réutilisés.

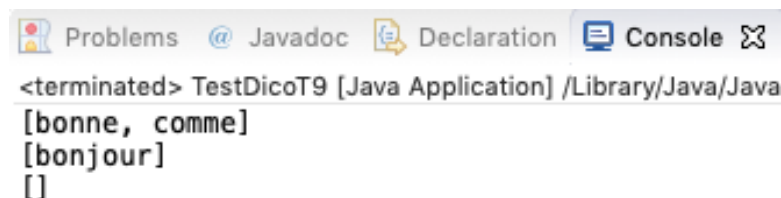
1. Créez une classe DicoT9 qui possède les méthodes suivantes :

- (a) **public void** enregistrer(String chaine) { ... }
- (b) **public** ArrayList<String> recuperer(String chaineT9){ ... }

Vous pouvez tester votre classe avec le programme suivant :

```
public class TestDicoT9 {
    public static void main(String[] args) {
        DicoT9 dico = new DicoT9();
        dico.enregistrer("bonjour");
        dico.enregistrer("bonne");
        dico.enregistrer("comme");

        System.out.println(dico.recuperer("26663"));
        System.out.println(dico.recuperer("2665687"));
        System.out.println(dico.recuperer("123456"));
    }
}
```

**Exercice II (Géométrie et POO, le retour)**

On souhaite manipuler des objets du plan. On peut réutiliser la classe Point et la classe Vecteur créées au TD précédent.

1. Créez une classe abstraite Figure qui représente une figure géométrique. Elle doit disposer des méthodes suivantes :
  - **public abstract double** perimetre();
  - **public abstract double** surface();
  - **public abstract** Figure translation(Vecteur v);
2. Créez une classe Disque qui hérite de Figure.<sup>1</sup>
3. Créez une classe abstraite Quadrilatere qui hérite de Figure. Un quadrilatère peut être caractérisé par ses quatre sommets, et son périmètre est facile à calculer : il s'agit de la somme des longueurs de ses quatre côtés. Implémentez cette classe, en déterminant quelles parties sont abstraites, et quelles parties ne le sont pas.

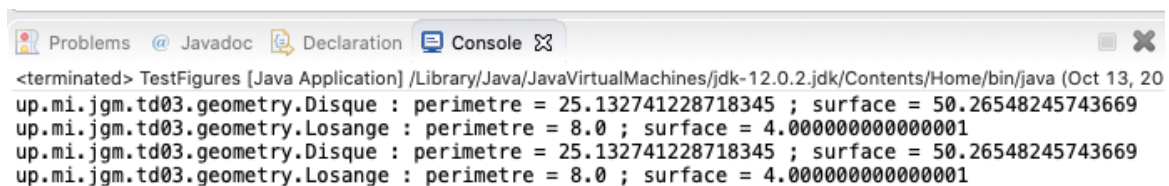
1. Vous pouvez vous inspirer de la classe Disque créée au TD précédent.

4. Un losange est un quadrilatère dont les quatre côtés ont la même longueur. Créez la classe Losange et implémentez les méthodes nécessaires.

Vos classes peuvent être utilisées avec le code suivant :

```
public class TestFigures {
    public static void main(String[] args){
        Disque d = new Disque(new Point(3.5, 4), 4);
        Losange l = new Losange(new Point(0, 0),
            new Point(2, 0), new Point(2, 2), new Point(0, 2));

        System.out.println(d);
        System.out.println(l);
        Vecteur v = new Vecteur(new Point(2,2), new Point(3,3));
        System.out.println(d.translation(v));
        System.out.println(l.translation(v));
    }
}
```



```
<terminated> TestFigures [Java Application] /Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java (Oct 13, 20
up.mi.jgm.td03.geometry.Disque : perimetre = 25.132741228718345 ; surface = 50.26548245743669
up.mi.jgm.td03.geometry.Losange : perimetre = 8.0 ; surface = 4.000000000000001
up.mi.jgm.td03.geometry.Disque : perimetre = 25.132741228718345 ; surface = 50.26548245743669
up.mi.jgm.td03.geometry.Losange : perimetre = 8.0 ; surface = 4.000000000000001
```

### Exercice III (Répertoires améliorés)

On a créé précédemment une classe RepertoireSimple qui permet :

- l'enregistrement d'une personne identifiée par son prénom, son nom et son numéro de téléphone,
- et la recherche d'un numéro de téléphone en connaissant l'identité de la personne.

On souhaite améliorer les répertoires avec de nouvelles fonctionnalités :

- un répertoire est associé à un contact particulier, qui est le propriétaire du répertoire ;
- on souhaite pouvoir rechercher l'identité d'une personne en fonction de son numéro de téléphone ;
- on souhaite pouvoir afficher l'ensemble des contacts contenus dans le répertoire : d'abord les informations sur le propriétaire du répertoire, puis tous les autres contacts triés par ordre alphabétique (du nom d'abord, du prénom ensuite).

Créez une classe RepertoireAmeliore, qui hérite de RepertoireSimple, et dont l'utilisation est illustrée dans le code suivant :

```
public class TestRepertoireAmeliore {
    public static void main(String[] args) {
        RepertoireAmeliore rep = new RepertoireAmeliore(
            new Personne("Jimi", "Hendrix", "0987654321"));
        rep.addPersonne("John", "Lennon", "0123456789");
        rep.addPersonne("Paul", "McCartney", "0234567891");
        rep.addPersonne("George", "Harrison", "0345678912");
    }
}
```

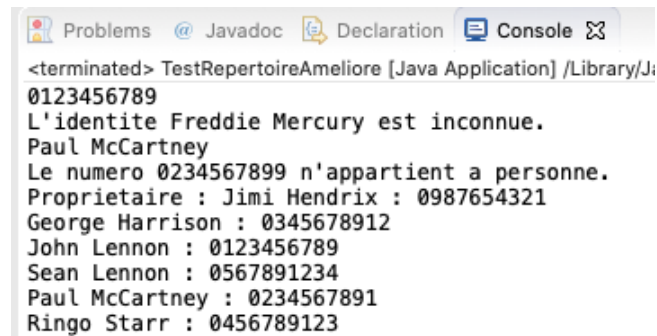
```

rep.addPersonne("Ringo", "Starr", "0456789123");
rep.addPersonne("Sean", "Lennon", "0567891234");

System.out.println(rep.chercheNumero("John", "Lennon"));
System.out.println(rep.chercheNumero("Freddie", "Mercury"));
System.out.println(rep.cherchePersonne("0234567891"));
System.out.println(rep.cherchePersonne("0234567899"));
System.out.println(rep);
}
}

```

L'exécution de ce programme donne le résultat suivant :



```

<terminated> TestRepertoireAmeliore [Java Application] /Library/J:
0123456789
L'identite Freddie Mercury est inconnue.
Paul McCartney
Le numero 0234567899 n'appartient a personne.
Proprietaire : Jimi Hendrix : 0987654321
George Harrison : 0345678912
John Lennon : 0123456789
Sean Lennon : 0567891234
Paul McCartney : 0234567891
Ringo Starr : 0456789123

```

#### Exercice IV (Produits et héritage)

Nous allons modéliser le fonctionnement d'une grande surface.

1. Le taux de TVA en France dépend de la nature du produit vendu. Il y a 4 taux différents : normal (20%), intermédiaire (10%), réduit (5,5%) et particulier (2,1%). Définissez une Enum qui permet de représenter les différents taux de TVA.
2. Un produit est une entité qui possède un prix hors taxes et un taux de TVA. Définissez une classe abstraite pour représenter un produit. Une méthode permet de calculer le prix TTC en fonction du prix HT et du taux de TVA.
3. Nous définissons maintenant des classes représentant différents types de produits :
  - (a) un DVD possède un titre, un réalisateur, et son taux de TVA est normal;
  - (b) un livre possède un titre, un auteur, un ISBN<sup>2</sup>, et son taux de TVA est réduit;
  - (c) un fruit possède un nom, un pays d'origine, et son taux de TVA est réduit;
  - (d) un médicament possède un nom, un laboratoire, le nombre de comprimés dans la boîte, et le taux de TVA est particulier.

#### Exercice V (POO et télévision)

La grille de programmation d'une chaîne de télévision est une suite de programmes de différents types :

- les émissions de divertissement,
- le journal télévisé,
- les reportages,

---

2. [https://fr.wikipedia.org/wiki/International\\_Standard\\_Book\\_Number](https://fr.wikipedia.org/wiki/International_Standard_Book_Number)

- les fictions.

Tous les programmes sont caractérisés par leur heure de début et de fin.<sup>3</sup> Les émissions de divertissement ont un présentateur, un nom, et durent toutes deux heures. Les fictions sont définies par leur titre, le nom du réalisateur, et s'il s'agit ou non d'une rediffusion. Le journal télévisé a un présentateur. Enfin, un reportage a un thème parmi trois possibles : histoire, actualité, culture) et un nom.

Définissez les classes qui permettent de modéliser les informations données ci-dessus.

### Exercice VI (POO et opérations mathématiques)

On peut représenter une opération mathématique sous forme d'un arbre dont les noeuds sont des opérateurs, et les feuilles sont des valeurs numériques. On souhaite représenter de telles expressions et pouvoir évaluer leur valeur.

- La classe (abstraite) `Operateur` représente aussi bien un noeud qu'une feuille de l'arbre. Un `Operateur` est caractérisé par son arité, qui est 0 dans le cas des feuilles, et un entier  $> 0$  pour les noeuds internes.
- Les opérateurs arithmétiques habituels (Somme, Soustraction, Multiplication, Division), d'arité 2, sont des classes filles d'`Operateur`.
- On peut également définir des classes `AdditionNAire` et `MultiplicationNAire` pour les versions générales de l'addition et la multiplication (l'arité est donc variable).
- Une `Valeur` a une arité nulle, et représente un nombre réel.

La classe `Operateur` contient une méthode abstraite **public abstract double** `evaluer()` ; qui retourne la valeur de cet opérateur pour ses opérands.

Implémentez ces classes.

---

3. Pour simplifier, les heures sont données sous forme de nombres entiers, entre 0 et 23.