	Université de Corse - Pasquale PAOLI
	Diplôme : **Licence SPI 3ème année** — 2023-2024
	UE : Ateliers de programmation
	**Programmation Orientée Objet** **Atelier 6 : Mise à niveau classes et Objets** - Création de classes simples - Principe d'encapsulation - Constructeur - Surcharge de constructeurs - Attributs et méthodes de classe - Recherche dans la documentation Java
	Enseignants : Paul-Antoine BISGAMBIGLIA, Marie-Laure NIVET, Evelyne VITTORI

Exercice 1 - **Robots**

Créez un nouveau package appelé ***exercice1*** dans votre projet Atelier1.

Question 1.1 Classe Robot

Définissez une classe Robot possédant :

- Un attribut de type Chaîne de caractères représentant la référence du robot,
- Un attribut de type Chaîne de caractères représentant le nom du robot,
- Deux attributs de type entier x et y décrivant la position courante du robot (coordonnées spatiales),
- Un attribut de type entier représentant l'orientation du robot : un robot peut en effet être orienté selon l'un des quatre points cardinaux NORD, SUD, EST ou OUEST.
- Un attribut comptabilisant le nombre total de robots créés,
- Un constructeur possédant quatre paramètres : le nom du robot, ses coordonnées de départ x,y et son orientation initiale. La référence du robot est générée automatiquement sous la forme ROB suivi du numéro du robot (par exemple : ROB1 pour le premier robot, ROB2 , ..ect…) ;
- Un constructeur possédant un seul paramètre (le nom du robot) et initialisant x et y aux valeurs par défaut classiques. La référence est générée automatiquement (ROB1, ROB2, …). La valeur par défaut de l'orientation est le nord.
- Une méthode de modification de l'orientation du robot admettant un paramètre de type entier représentant l'orientation souhaitée,
- Une méthode déplacer permettant au robot d'avancer d'une unité. Selon l'orientation du robot, ce déplacement consistera à ajouter ou retrancher 1 à son abscisse ou son ordonnée : *Par exemple, si le robot est orienté vers le nord, il avancera en ajoutant 1 à son ordonnée. S'il est orienté vers l'Est, il ajoutera 1 à son abscisse.*

  Vous prendrez soin de vérifier que les coordonnées ne deviennent pas négatives !

Indication

- Les caractéristiques des robots sont illustrées sur la figure 1.
- Les points cardinaux pourront être représentés par des constantes entières placées dans la classe Robot : NORD=1, EST=2, SUD=3, OUEST=4.

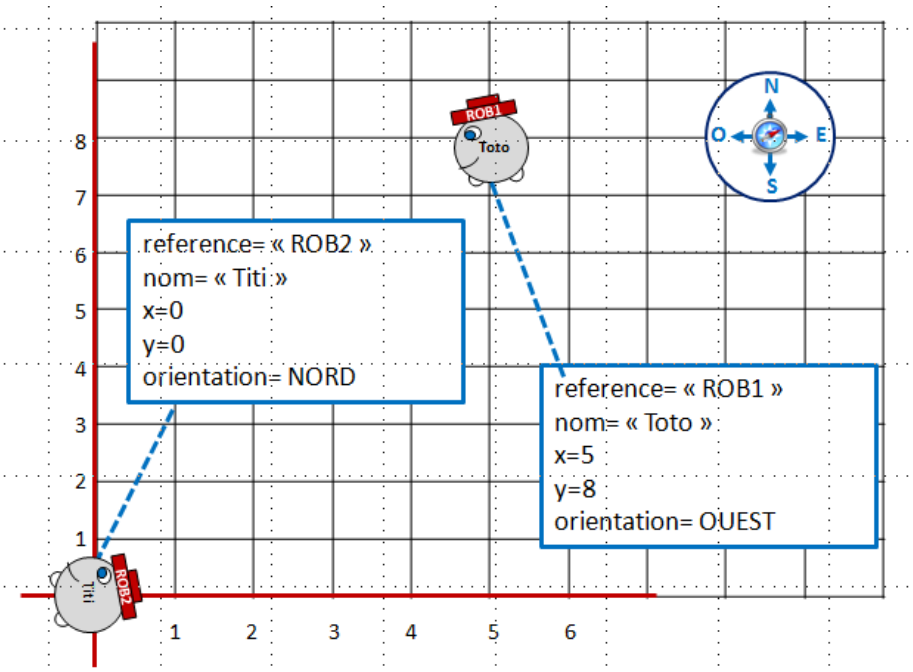

Figure 1 : Caractéristiques des robots

Question 1.2 Classe ManipRob

☞ Définissez une classe ManipRob possédant une méthode main exécutant les actions suivantes :

- création de deux objets de la classe Robot :
  - Toto ayant pour point de départ (10, 20) direction Sud
  - Titi ayant pour point de départ (0,0) direction Nord
- déplacements des robots avec changements de direction

Question 1.3 Affichage des Robots

☞ Dans la classe Robot, ajoutez une méthode afficheToi() qui affiche une description du robot (nom, référence, coordonnées et direction). Utilisez afficheToi() dans la méthode main() de ManipRobot.

☞ Dans la méthode main() de la classe ManipRobot, ajoutez l'instruction System.out.println(rob) où rob désigne un des robots que vous avez créés.

L'affichage obtenu correspond en fait au résultat renvoyé par l'exécution de la méthode toString() qui est une méthode « héritée » de la classe Object (la racine de toutes les classes Java). Tous les objets possèdent cette méthode. Pour l'instant, on peut simplement constater que l'affichage que l'on obtient ne nous intéresse pas trop !! On souhaiterait obtenir l'affichage

des caractéristiques du robot. Pour obtenir cela, il nous suffit de définir notre propre version de la méthode toString.

Vous allez donc redéfinir cette méthode afin qu'elle renvoie une chaîne de caractères qui décrit le robot.

Attention votre méthode devra bien respecter la signature suivante :

```
public String toString(){  …….. }
```

 Exécutez à nouveau la classe ManipRobot. Voyez ce qui est affiché maintenant par l'instruction System.out.println(rob).

Vous pouvez ainsi constater que println() utilise automatiquement la méthode toString() de la classe de l'objet qu'il a à imprimer.

 Modifiez le main de la classe ManipRobot afin de ne plus utiliser la méthode afficheToi mais directement System.out.println pour afficher les caractéristiques des robots.

Exercice 2 - **Vecteurs3D**

Question 2.1 Classe Vecteur3D

Définissez une classe Vecteur3D permettant de manipuler des vecteurs à trois composantes de type réel (x, y et z) et disposant :

- d'un constructeur à trois arguments,
- d'un constructeur sans argument qui crée un vecteur de la forme (0,0,0),
- d'une méthode d'affichage des coordonnées du vecteur sous la forme :
  <x, y, z>
- d'une méthode sans paramètre qui a pour résultat un nombre réel représentant la norme du vecteur,
- d'une méthode produitScalaire ayant pour résultat un nombre réel représentant le produit scalaire de deux vecteurs,
- d'une méthode somme ayant pour résultat un vecteur3D représentant la somme de deux vecteurs.

Remarque : vous définirez deux versions différentes (et équivalentes) de chacune des méthodes produitScalaire et somme :

- *une méthode d'instance comportant un paramètre*
- *une méthode statique("de classe") comportant deux paramètres*

Question 2.2 Classe test

Définissez une classe de test créant successivement deux vecteurs affichant leur coordonnées, leurs normes respectives, leur produit scalaire et leur somme.

Exemple d'exécution:

```
v1 = < 3.0, 2.0, 5.0 >
Norme de v1 = 6,16
v2 = < 1.0, 2.0, 3.0 >
```

```
Norme de v2 = 3,74
v1 + v2 = < 4.0, 4.0, 8.0 >
v1.v2 = 22.0
```

Rappels :

Soient deux vecteurs v et w ayant respectivement pour coordonnées (x,y,z) et (x',y',z'),

- *la norme de v est donnée par la racine carrée de* $(x^2 + y^2 + z^2)$
- *le produit scalaire v.w est un réel égal à (x*x' + y*y' +z*z')*
- *la somme v+w est un vecteur ayant pour coordonnées (x+x', y+y', z+z')*

Indications

- racine carrée: méthode statique de la classe Math sqrt
- Pour afficher un réel avec uniquement 2 décimale, il faut créer un objet Decimal Format

```
DecimalFormat df=new DecimalFormat("#0.00");
```

et ensuite utiliser cet objet pour afficher les variables de type réel.
Par exemple pour afficher une variable x de type réel, on écrira:

```
System.out.println(df.format(x));
```

Exercice 3 - Manipulation de Classes de l'API Java

Question 3.1 classe Math

Recherchez dans la documentation java (http://download.oracle.com/javase/8/docs/api/), la classe Math du package java.lang et répondez aux questions suivantes :

1. Combien d'attributs possède cette classe ? aucun attribut
2. Quelle est la particularité des attributs et des méthodes de cette classe ? elle fournit un ensemble de méthodes statiques pour effectuer des opérations mathématiques courantes
3. Identifiez la méthode permettant de générer un nombre aléatoire compris entre 0 et 1. Donnez sa signature et indiquez s'il s'agit d'une méthode de classe ou d'instance.
4. Combien il y a-t-il de méthodes nommées « max » ? Donnez leurs signatures et expliquez leurs différences. 8 méthodes. Ces méthodes "max" permettent de comparer des valeurs de différents types numériques et de retourner la valeur maximale parmi celles-ci
5. On considère les instructions suivantes:
   a) int x= Math.max(5) ; La méthode Math.max nécessite au moins deux arguments
   b) int x= Math.max(5,6) ; Cette instruction est correcte au niveau syntaxique . x=6

Pou chacune de ces instructions, indiquez si elle correcte ou non au niveau syntaxique. Si elle n'est pas correcte, expliquez la nature de l'erreur et si elle est correcte, indiquez la valeur finale de la variable x.

Question 3.2 classe String

Recherchez dans la documentation java, la classe String du package java.lang et répondez aux questions suivantes :

1. Recherchez la méthode compareTo. Donnez sa signature et précisez s'il s'agit d'une méthode de classe ou d'instance. Il s'agit d'une méthode d'instance, type de retour Int
2. On considère l'instruction suivante:
   String res= String.compareTo("bonjour") ; il faut un instance. Le type de retour c'est int et non String

Cette instruction n'est pas correcte pour au moins deux raisons, lesquelles ?

3. Recherchez la documentation relative à la méthode length. Donnez sa signature et précisez s'il s'agit d'une méthode de classe ou d'instance. Il s'agit d'une méthode d'instance. Type de retour int
4. .On considère la séquence d'instructions suivantes:
   String st=("bonjour")
   int lg= String.length(st) ;

On souhaite affecter à la variable lg, le nombre de caractères de la chaine st.
La deuxième instruction n'est pas correcte, expliquez pourquoi et proposez une solution pour la corriger. lenght méthode d'instance il faut instancier par objet. Lenght n'admet aucun attribut
int lg= st.length() ;

Question 3.3 **Définissez une classe testAPI comportant une méthode main réalisant les actions suivantes :**

1. Afficher le nombre PI
2. Afficher un nombre réel aléatoire compris entre 0 et 1 (exclu).
3. Afficher un nombre entier aléatoire compris entre 1 et 3 (inclus).
4. Définissez deux variables entières x1 et x2, initialisez-les et affichez la plus grande des deux en invoquant une méthode max de la classe Math.
5. Définissez deux variables String n1 et n2, initialisez les et affichez la première selon l'ordre alphabétique en invoquant la méthode compareTo de la classe String.

Indications :

1- Pour obtenir la partie entière d'un nombre réel de type double, il suffit de le convertir en int en utilisant un « cast ». Exemple :

```
double r=2.245566;
int convR= (int) r; //r sera égal à 2
```

2- Pour afficher un nombre réel avec uniquement deux chiffres dans la partie décimale, une solution consiste à définir un objet de la classe DecimalFormat en précisant le format d'affichage et à invoquer ensuite la méthode format sur cet objet pour obtenir une chaine de caractère formatée :

Exemple :

```
double x=2.2455666;
DecimalFormat df=new DecimalFormat("#0.00");
System.out.println(df.format(x));
```