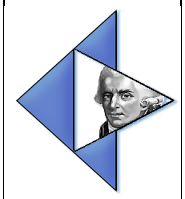
	Université de Corse - Pasquale PAOLI
	Diplôme : **Licence SPI 3ème année** | 2024-2025
	UE : Ateliers de programmation
	**Programmation Orientée Objet** **Atelier 1 : Mise à niveau POO** - Création de classes simples - Principe d'encapsulation - Constructeur - Surcharge de constructeurs - Attributs et méthodes de classe - Recherche dans la documentation Java - Héritage, constructeurs et redéfinition de méthode - Classes abstraites et polymorphisme
	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.

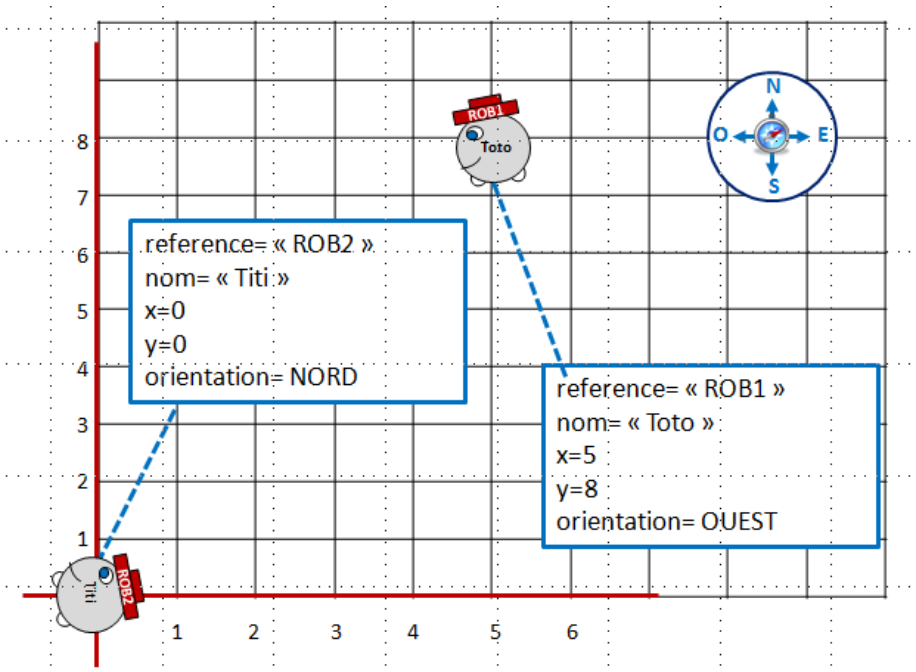


Figure 1 : Caractéristiques des robots

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 ?
2. Quelle est la particularité des attributs et des méthodes de cette classe ?
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.
5. On considère les instructions suivantes:
   a) int x= Math.max(5) ;
   b) int x= Math.max(5,6) ;

   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.
2. On considère l'instruction suivante:
   String res= String.compareTo("bonjour") ;

   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.
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.

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));
```

Exercice 4 - Héritage, constructeurs et redéfinitions

Question 4.1 Récupérez sur Teams le code de la classe Personne (Personne.java).

Attention ! Ce code ne devra pas être modifié dans les questions suivantes.

Question 4.2 Définissez le code d'une classe Etudiant (avec un attribut de type String numEtu) et d'une classe Enseignant (attributs salaire et nbheuresCours) sans définir de constructeur ni ajouter de méthode autre que setNumEtu dans Etudiant.

Question 4.3 Créez une classe TestPersonne avec un main réalisant les actions suivantes:

- Créer une Personne pers de nom Marie et d'age 20 ans et l'afficher
- Créer un Etudiant etu (vide) et l'afficher
- Lui affecter le nom Pierre, l'age 21 et le numEtu « 20203456 »
- Afficher à nouveau etu

Vous devez obtenir l'affichage suivant :

```
Nom : Marie
Age : 20
Nom :
Age : 0
Nom : Pierre
Age : 21
```

Question 4.4 Récupérez sur Teams le code de la classe TestPersonneBis (TestPersonneBis.java).

Question 4.5 Complétez vos classes Etudiant et Enseignant afin que le programme de test TestPersonneBis provoque l'affichage suivant :

```
Marie (20 ans)
Etudiant numero 20203433 Jean (21 ans)
Enseignant Pierre (54 ans) 3000.0 euros
Nom : Marie
Age : 20
*******Etudiant*******
Numero étudiant : 20203433
Nom : Jean
Age : 21
*******Enseignant******
Nom : Pierre
Age : 54
```

```
Salaire : 3000.0
Nombre d'heures de cours : 250
```

Exercice 5 - Classes abstraites et polymorphisme

Les salaires des enseignants d'une université sont calculés de manière spécifique en fonction de leur catégorie :

- les titulaires qui ont un salaire fixe
- les vacataires ont un salaire qui dépend du nombre d'heures de cours qu'ils assurent sachant qu'une heure de cours est rémunérée 40 euros.

Question 5.1 Définissez une classe abstraite Enseignant comportant un attribut nom de type String, un constructeur à un paramètre et une méthode abstraite *salaire()* renvoyant un double et une méthode getNom.

Question 5.2 Définissez une classe Titulaire héritant de Enseignant comportant un attribut salaire de type double, un constructeur à deux paramètres et une implémentation de la méthode salaire renvoyant son salaire.

Question 5.3 Définissez une classe Vacataire héritant de Enseignant comportant une constante de type double égale à 40, un attribut nbHeuresCours de type int, un constructeur à deux paramètres et une implémentation de la méthode salaire renvoyant son salaire calculé à partir de son nombre d'heures de cours.

Question 5.4 Définissez une classe Université comportant :

- un attribut nom de type String,
- un attribut listeEnseignants de type ArrayList<Enseignant>,
- un constructeur à un paramètre,
- une méthode afficherSalaires qui affiche à l'écran la liste des noms des enseignants avec leurs salaires ainsi que le total des salaires versés
- une méthode embaucher qui comporte un paramètre e de type Enseignant et ajoute l'enseignant e à la liste d'enseignants listeEnseignants.

Question 5.5 Définissez un programme de test afin d'obtenir l'affichage suivant :

```
LISTE DES ENSEIGNANTS DE l'UNIVERSITE Pascal Paoli
Effectif: 4 enseignants
Pierre (titulaire)  : 1500.0 euros
Laurent (titulaire)  : 2500.0 euros
Michel (vacataire 15 heures) : 600.0 euros
Marie (vacataire 60 heures) : 2400.0 euros
Total des salaires = 7000.0 euros
```

Prenez soin de redéfinir judicieusement les méthodes toString de vos classes Enseignants, Titulaire et Vacataire afin d'éviter la duplication de code.

Exercice 6 - Classes abstraites et polymorphisme

On considère les classes Personne, Etudiant et Enseignant de l'exercice 4.
Identifiez les erreurs d'exécution dans les programmes suivants et expliquez-les en ajoutant des commentaires :

```
public class TestPersonne2 {
public static void main(String[] args) {
Personne p1=new Personne("Marie",20);
Personne p2=new Etudiant("Machin",26,"20202134");
Personne p3=new Enseignant("Toto",34,2000,192);
Etudiant p4=new Etudiant("Jean", 21,"20203433");
Enseignant p5= new Enseignant("Pierre",54,3000,250);
p4=(Etudiant) p2;
```

```
p5=(Enseignant) p1;
((Etudiant) p2).setNumEtu("20205784");
p4.setNumEtu("20205785");
((Etudiant) p3).setNumEtu("20205786");
}
```

```
public class TestPersonne3 {
public static void main(String[] args) {
Personne p1=new Personne("Marie",20);
Personne p2=new Etudiant("Machin",26,"20202134");
Personne p3=new Enseignant("Toto",34,2000,192);
Etudiant p4=new Etudiant("Jean", 21,"20203433");
Enseignant p5= new Enseignant("Pierre",54,3000,250);
p4=(Etudiant) p2;
p5=(Enseignant) p1;
((Etudiant) p2).setNumEtu("20205784");
p4.setNumEtu("20205785");
((Etudiant) p3).setNumEtu("20205786");
}
```