# TP n°3 – Les Objets

Projet de programmation (IK2)

## Les Objets

**Exercice 1 — *Vecteurs de l'espace*** Réaliser une classe `Vecteur3d` permettant de manipuler des vecteurs à trois composantes (de type `double`) et disposant :

1. D'un constructeur qui ne prend pas d'argument
2. D'un constructeur à trois arguments.
3. D'une méthode d'affichage des coordonnées du vecteur.
4. D'une méthode fournissant la norme euclidienne du vecteur (rappel : $||(x, y, z)|| = \sqrt{x^2 + y^2 + z^2}$).
5. D'une méthode fournissant la somme de deux vecteurs.
6. D'une méthode fournissant le produit scalaire de deux vecteurs (rappel : $\langle\ (x, y, z)\ |\ (x', y', z')\ \rangle\ =\ xx'\ +\ yy'\ +\ zz'$).
7. D'une méthode fournissant une copie d'un vecteur.
8. D'une méthode qui renvoie `true` si deux vecteurs sont égaux.
9. D'une méthode qui renvoie `true` si deux vecteurs sont différents.
10. D'une méthode fournissant le vecteur normalisé du vecteur.

Au fur et à mesure, écrire un programme `TestVecteurs` mettant en œuvre ces méthodes.

**Exercice 2 — *Nombres complexes***

1. Réaliser une classe `Complexes` permettant de manipuler les nombres complexes. Outre un constructeur, on écrira notamment des méthodes permettant de :
   (a) Afficher un nombre complexe.
   (b) Dessiner un nombre complexe (dans un repère orthonormé, d'unité 10 pixels).
   (c) Calculer la somme et le produit de complexes.
   (d) Calculer la division d'un complexe par un réel
   (e) Déterminer le module (à l'aide de `Math.sqrt`) et l'argument (à l'aide de `Math.atan` et de la formule $arg(z) = arctan(y/x)$) d'un complexe.
   (f) Effectuer les transformations usuelles (rotation $r(z, \theta) = ze^{i\theta}$, translation).
2. Éventuellement à l'aide du premier point, effectuer une méthode permettant de dessiner un polygone régulier. Le centre, le rayon et le nombre de côtés seront entrés par l'utilisateur.
3. On souhaite approximer la fonction exponentielle en utilisant son développement en série entière :

$$e^z = \sum_{i=0}^{+\infty} \frac{z^i}{i!}$$

Écrire une méthode `exponentielle`, qui, pour un `complexe z` et un `int n`, retourne la somme :

$$\sum_{i=0}^{n} \frac{z^i}{i!}$$

4. Écrire une méthode qui calcule le cosinus et le sinus d'un angle $\theta$ en utilisant le développement limité de $e^{i\,\theta}$ à l'ordre $n$ (on passera donc, en plus de l'angle `theta`, l'entier `n` en argument). On rappelle que :
$$e^{i\,\theta} = \cos\theta + i\,\sin\theta$$
5. Tester cette méthode pour différentes valeurs de `theta` et `n`. Vérifier qu'avec $n = 15$ et $\theta = \frac{\pi}{6}$, on obtient une bonne approximation des valeurs du cosinus ($\frac{\sqrt{3}}{2} \approx 0.866025404$) et du sinus ($\frac{1}{2}$).

**Exercice 3 — *Machine à café*** Le but de cet exercice est d'écrire le contrôleur du monnayeur d'une machine à café.

La machine que nous voulons simuler permet d'obtenir soit un café, un thé, un chocolat ou un potage tomate sélectionnable par des boutons situés sur le panneau avant de la machine. Chaque consommation coûte 50 centimes qu'il faut ajouter dans le monnayeur avant de sélectionner sa boisson. La machine a également un bouton d'annulation qui permet de récupérer la monnaie insérée avant la sélection.

Le contrôleur du monnayeur gère la caisse de la machine et un compteur qui indique la somme qui a été ajoutée dans la machine depuis le service de la dernière boisson. Le contrôleur vérifie également que l'utilisateur a mis suffisamment d'argent dans la machine avant de servir une boisson.

1. Définir une classe `Monnayeur` qui a les champs `caisse` et `compteur` et un constructeur qui permet de les initialiser. Puis écrire une méthode `etat` qui affiche l'état de la machine (les valeurs de `caisse` et de `compteur`) Définir une classe `Main` qui crée un objet de type `Monnayeur` et affiche son état.
2. Écrire la méthode qui permet d'ajouter de l'argent dans la machine. La somme à ajouter sera donnée par un paramètre de type double.
3. Écrire la méthode qui correspond au bouton d'annulation.
4. Écrire la méthode qui vérifie si l'utilisateur a mis suffisamment d'argent dans la machine pour avoir une boisson. Cette méthode retourne une valeur supérieure ou égale a zéro si la machine peut servir la boisson et négative sinon. La valeur absolue de la valeur retournée correspond à la monnaie rendue ou à ce qu'il faut ajouter pour avoir une boisson.
5. Écrire un programme qui gère deux machines à café.
6. Modifier la machine pour qu'elle n'accepte que les pièces de 20 et 50 centimes et de 1 euros. La machine ne pourra rendre la monnaie que si elle a les bonnes pièces dans sa caisse.

**Exercice 4 — *Un peu de hasard*** – Écrire une classe De avec un attribut val. Initialisé ce dernier à 1 dans le constructeur de la classe.

- Surcharger le constructeur en initialisant val à une valeur donnée en paramètre.
- Écrire la méthode modifiant l'attribut val à une valeur aléatoire comprise entre 1 et 6.
- Écrire une méthode dessinant un dé (un carré contenant 9 cercles).
- Écrire un programme qui lance deux dés et affiche la somme puis la valeur de chaque dé. Dessiner les deux dés.
- Écrire un programme qui lance trois dés et les dessine. Á chaque étape choisir un seul dé pour être relancé et redessiné. Le jeu s'arrête lorsque les trois valeurs 135 apparaissent.

**Exercice 5 — *Bouteille*** Afin de stocker du liquide, on est amené à définir le concept de bouteille. Chaque bouteille a une contenance maximale et contient une certaine quantité de liquide (volume courant). Lors de la création d'une bouteille sa capacité maximale est donnée, ainsi que si elle ferme à vis ou non. La bouteille peut être créée vide ou contenant une quantité initiale. On veut pouvoir :

- remplir partiellement la bouteille d'une certaine quantité,
- la remplir complètement,

- la vider d'une certaine quantité,
- la vider complètement,
- connaître sa contenance maximale,
- afficher ses caractéristiques (contenance, quantité actuelle, fermeture à vis ou non).

Bien évidemment, on ne peut pas remplir plus que la contenance de la bouteille, ni vider d'une quantité supérieure à la quantité actuelle. En cas de remplissage avec une quantité plus grande que ne peut contenir la bouteille, celle-ci sera remplie à son maximum possible.
Écrire en Java la classe Bouteille en respectant la description ci-dessus.