# Développement d'applications réparties (rappel Java) : Série 1

**Exercice 1 :**

Le **Bandit Manchot** est un jeu de hasard que l'on joue avec une machine comportant quatre roues qui tournent et qui comportent chacune un symbole (par exemple abricot, banane, cerise, datte, étoile). Quand les roues s'arrêtent, si 4 roues ont un symbole identique on a gagné.

**On se propose :**

De réaliser ce jeu de hasard en mode texte. On figurera les symboles par un chiffre ou par une lettre. Les gains possibles seront les suivants : 10 € pour deux symboles identiques, 100 € pour 3 symboles identiques, 1000 € pour 4 et 0 € sinon.

Créer la classe Roue pour le choix des symboles avec les méthodes suivantes:

- public void lance() : Génère un nombre aléatoire et le stocke dans un attribut figurant la valeur affichée sur la roue.
- public int get() : renvoie la valeur stockée

Créer une classe BanditManchot qui comporte un tableau de 4 roues, un constructeur pour créer ce tableau et une méthode public int joue() qui affichera les 4 roues et renverra le gain au jeu: 10 € si deux symboles identiques, 100 € si 3 symboles identiques, 1000 € pour 4 symboles sont identiques et 0 € sinon.

Ecrire un programme TestBanditManchot.java qui contient une méthode main qui permet de tester les classes crées.

**Exercice 2 :**

Pour gérer des notes d'étudiants dans diverses matières, on désire écrire les classes suivantes :

- Une classe Etudiant : Un étudiant possède un nom et une adresse.
- Une classe ListEtudiant : Chaque étudiant est représenté par un objet unique, maintenu dans un tableau, le tableau de tous les étudiants.
  On doit pouvoir ajouter un étudiant à la liste des étudiants, modifier l'adresse d'un étudiant, obtenir un étudiant à partir de son nom.
- Une classe Matiere : Pour chaque matière, certains étudiants possèdent une note. Pour une matière donnée, on doit pouvoir ajouter un étudiant et sa note, demander la note d'un étudiant, modifier la note d'un étudiant, calculer la moyenne des notes, imprimer la liste des notes avec le nom et l'adresse des étudiants.

Ecrire un programme qui teste ces classes.

**Exercice 3 :**

L'exercice suivant est un algorithme de tri générique. La procédure **AlgoGene.tri** tri un tableau d'objets de type quelconque. La seule propriété exigée est que les éléments soient comparables au moyen de la procédure **inf** (Object x, Object y) qui rend vrai si x est inférieur

(en un certain sens) à y. Pour cela on utilise une interface **Compare** qui définit le profil de la procédure **inf**. La procédure **tri** reçoit en paramètre un objet comparateur **op** qui implémente cette interface.

```
interface Compare {
        public boolean inf(Object x, Object y);
}
class AlgoGene {
public static void tri (Object [] T, Compare op) {

        for (int i=T.length-1; i>=0; i--) {
                for (int j=l; j<=i; j++) {
                        if (op.inf(T[j],T[j-1])) {
                                Object x=T[j-1]; T[j-1]=T[j]; T[j]=x
        }}}
}}
```

Pour trier n'importe quel type d'objets selon n'importe quel critère, il suffit d'implémenter l'interface **Compare** au moyen d'une classe convenable et de passer en paramètre du tri une instance de cette classe. C'est ce qui est proposé dans l'exercice suivant.

Des personnes sont caractérisées par un nom, un âge et un poids:

```
class Personne.{
        String nom;
        int age;
        int poids;
        public Personne(String n, int a, int p){
                nom=n; age=a; poids=P;
        }
}
```

On définit la population suivante:

```
Personne [] peuple = new Personne[4];
peuple [0] = new Personne("toto", 25, 80);
peuple [1]= new Personne(("tutu", 53, 65);
peuple [2] = new Personne(("tata", 15, 47);
peuple [3]= new Personne(("jojo", 12, 30);
```

Utiliser **AlgoGene** pour trier cette population selon leur âge puis selon leur poids.