# L3 MIAGE – Java avancée – TP3

## Modalités du rendement

- Envoyez juste les fichiers .java de l'exercice.
- Mettez les fichiers de chaque exercice dans un dossier nommé *ex[numéro de l'exercice]*
- Mettez tous les dossiers des exercices dans un dossier qui porte le nom suivant : *[votre_prénom]_[votre_nom]_TP[numéro_de_tp]*
- Zippez le dossier sans changement de nom ; finalement envoyer un fichier zip de format : *[votre_prénom]_[votre_nom]_TP[numéro_de_tp].zip*

## Exercice 1

Ajouter les commentaires nécessaires pour expliquer le fonctionnement du code suivant (copier le code dans votre IDE) :

Exécuter le code ; Est-ce que tous les résultats sont affichés à la fin de code ? Que faut-il modifier dans le code pour que toutes les threads parviennent à afficher leurs résultats ?

```java
import java.util.*;
import java.util.stream.*;

public class Thrds {

	public static class Divisible extends Thread {
		private int divisiblePar = 2;
		public List<Integer> list = new ArrayList<Integer>();

		public Divisible(ThreadGroup divs, String name, int div) {
			super(divs, name);
			this.divisiblePar = div;
		}
		public void run() {
			List<Integer> dlist = new ArrayList<Integer>();
			for(Integer l:list) {
				if(l % divisiblePar == 0){
					dlist.add(l);
				}
			}

			System.out.println(this.getName() + " est terminé :" + dlist);
		}
	}

	public static void main(String[] args) {
		List<Integer> list2 = IntStream.
				range(1, 11).
				boxed().
				collect(Collectors.toList());
		List<Integer> list3 = IntStream.
				range(1, 1001).
				boxed().
				collect(Collectors.toList());
		List<Integer> list5 = IntStream.
				range(1, 100001).
				boxed().
				collect(Collectors.toList());
		ThreadGroup divs = new ThreadGroup("Groupe de division");
		Divisible div2 = new Divisible(divs, "div2", 2);
		Divisible div3 = new Divisible(divs, "div3", 3);
		Divisible div5 = new Divisible(divs, "div5", 5);

		div2.list = list2;
		div3.list = list3;
		div5.list = list5;

		div2.start();
		div3.start();
		div5.start();

		while (div2.isAlive() && div3.isAlive() && div5.isAlive()) {
			try {
				Thread.sleep(1);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
		divs.stop();
		System.out.println("Division terminée");
	}
}
```

# Exercice 2

## Mapping

1. Créez un thread « **Mapping** », qui contient deux variables :
   - **input_string** de type String
   - **output_map** de type Map<String, Integer>.

2. Ecrivez la méthode **run()** du thread qui applique les opérations suivantes sur la variable **input_string** :
   - Remplace toutes les occurrences de **virgule** « **,** » par **espace+virgule**.
   - Remplace toutes les occurrences du symbole **apostrophe** « **'** » par un **espace**.
   - Remplace toutes les occurrences du symbole **point** « \\. » par **espcae+point** (dans les commandes de remplacement, il faut toujours ajouter \\ avant le symbole point afin d'informer le compilateur qu'il ne s'agit pas d'une expression régulière).
   - Utilise la méthode **split(String regex)** pour découper la variable (après les remplacement) en mots séparés par **espaces**, et stocke le résultat dans une variable de type **String[]**.
   - Utilise la matrice des mots pour remplir la variable output_map dont les clés (string) correspondent aux mots, et les valeurs (integer) correspondent à leurs fréquences. Autrement dit, chaque entrée de la map contient un mot avec son nombre d'occurrences dans le texte originale.
   - Supprime la clé " " (espace).

## Reduce

3. Créez le thread **Reduce** qui possède deux variables :
   - **input_list** : de type ArrayList<Map<String, Integer>>
   - **output_map** : de type Map<String, Integer>
4. Ecrivez la méthode **run()** du thread qui calcule la somme d'occurrences d'un mot dans l'ensemble des maps de la liste input_list :
   - Faite une boucle for sur la liste input_list où **m<k, v>** et une map membre de liste
     - Itérez sur les entrées de m<k, v> :
       - Si output_map ne contient pas une clé=k, on lui ajoute l'entrée <k,v>
       - Si output_map contient déjà une éntré=<k,u>, on mets à jour la valeur de l'entrée pour qu'elle devient <k, v+u>

## Programme Mapping + Reduce

5. Ecrivez un programme qui a trois variables de chaines de caractères :
   - *Un jeune et timide 'Hobbit', Frodon Sacquet, hérite d'un anneau magique. Bien loin d'être une simple babiole, il s'agit d'un instrument de pouvoir absolu qui permettrait à Sauron, le 'Seigneur des ténèbres', de régner sur la 'Terre du Milieu' et de réduire en esclavage ses peuples. Frodon doit parvenir jusqu'à la 'Crevasse du Destin' pour détruire l'anneau.*
   - *Après la mort de Boromir et la disparition de Gandalf, la 'Communauté' s'est scindée en trois. Perdus dans les collines d''Emyn Muil', Frodon et Sam découvrent qu'ils sont suivis par Gollum, une créature versatile corrompue par l'anneau magique. Gollum promet de conduire les 'Hobbits' jusqu'à la 'Porte Noire' du 'Mordor'. A travers la 'Terre du Milieu', Aragorn, Legolas et Gimli font route vers le 'Rohan', le royaume assiégé de Theoden.*
   - *Les armées de Sauron ont attaqué 'Minas Tirith', la capitale de 'Gondor'. Jamais ce royaume autrefois puissant n'a eu autant besoin de son roi. Cependant, Aragorn trouvera-t-il en lui la volonté d'accomplir sa destinée ? Tandis que Gandalf s'efforce de soutenir les forces brisées de Gondor, Théoden exhorte les guerriers de Rohan à se joindre au combat. Cependant, malgré leur courage et leur loyauté, les forces des Hommes ne sont pas de taille à lutter contre les innombrables légions d'ennemis.*
6. Créez trois instants du thread **Mapping** dont chacune traite un des trois textes en 5.
7. Créez un instant du thread **Reduce** qui traite les 3 maps résultantes en 6, et affichez la map générée par ce thread.
8. Afficher le résultat de Reduce ordonné par la fréquence décroissante des mots .