TP 1 : Programmation parallèle avec JAVA

Consigne :

- Le compte rendu est à envoyer sur Edmodo avant la data limite qui vous sera indiqué par la plateforme.
- Tout retard est pénalisant.
- Précisez la configuration logicielle et matérielle de votre machine.
- Code commenté et capture de l'exécution sont requis.

Exercice 1 :

Vous disposez de deux programmes dans le répertoire exo 1 accompagnant cet énoncé.

1) Compiler, corriger si nécessaire.
2) Tester. Que fait ce programme ?
3) Répondez aux questions suivantes :

a. Quels sont les attributs d'un Thread ?
b. Quelles sont les valeurs de la priorité maximale, normale et minimale d'un thread ?
c. Quelle est la priorité des threads lancés ?
d. Quel est le format de nommage par défaut d'un thread ?
e. Quels sont les états d'un thread ?
f. Peut-on modifier l'ID ou l'état d'un thread ?
g. A quel groupe appartiennent les threads crées dons ce programme ?

Exercice2 :

Soit un programme exemple1.java, dont le code est le suivant :

```
public class exemple1 extends Thread {
      private String Salutation;
      private int attente;
      private int number;
public exemple1(String Salutation) {
      this.Salutation=Salutation;
}
public void run() {
      for (int i =1; i< 1000; i++)
      System.out.println(i + " " + Salutation );
}
public static void main(String args[]){
      exemple1 thread1, thread2, thread3;
      thread1=new exempleThread1("Bonjour ");
      thread2=new exempleThread1("Bonsoir ");
```

```
        thread3=new exempleThread1("à demain ");
        System.out.println("Je suis le main :)");
        thread1.start();
        thread2.start();
        thread3.start();
        System.out.println("main terminé");
        //System.exit(0);
        }
}
```

4) Rediriger la sortie vers un fichier log.
5) Que fait ce programme ?
6) Exécuter. Que constatez-vous ?
7) Décommenter l'instruction System.exit(0). Cette dernière est appelé par quel thread ?
8) Exécuter. Que constatez-vous ?
9) Modifier votre programme en ajoutant trois appels à la méthode *join*. Compiler et exécuter le programme. Que constatez-vous ?
10) Exposer une conclusion détaillée en exploitant vos constations

Exercice 3

Le but de cet exercice est de créer un gestionnaire de tâches. Ce dernier sera mis en œuvre par 2 types de thread. Un thread classique nommé writerTask pour stocker les tâches dans une structure de données de type ConcurrentLinkedQueue. Un deuxième thread Démon nommé cleanerTask dont la vocation est de traiter les tâches stockées dans la structure de données partagée. CleanerTask traite la dernière tâche dans la queue, si celle-là est ancienne (création date de plus de 10 secondes) elle sera supprimée. Après chaque suppression le thread démon affichera la nouvelle taille de la queue.

Pour ce faire vous aller réutiliser la classe Event suivante :

```
import java.util.Date;
public class Event {
      private Date date;
      private String event;
public Date getDate() {
      return date;
}
public void setDate(Date date) {
      this.date = date;
}
public String getEvent() {
      return event;
}
public void setEvent(String event) {
      this.event = event;
}}
```

1. Continuer l'implémentation des classes suivantes :

```
import java.util.Date;
import java.util.Deque;

public class CleanerTask extends Thread {
private Deque<Event> deque;
```

```
	public void run() {
		while (true) {
			Date date = new Date();
			clean(date);
		}
	}
	private void clean(Date date) {

	}
	}

	import java.util.Date;
	import java.util.Deque;
	import java.util.concurrent.TimeUnit;
	public class WriterTask implements Runnable {
	private Deque<Event> deque;

	public WriterTask(Deque<Event> deque) {
		this.deque = deque;
	}
		public void run() {

		// Writes 100 events in deque


	}
```

1. Tester vos classes en implémentant un programme principal qui instancie autant de writerThread que de cœurs dans votre machine et un seul démon.