Le patron *Singleton*

Exercice 1 – Codage d'un singleton

On souhaite permettre à une application donnée de pouvoir enregistrer un journal d'évènements à des fins de déverminage (debugging). Pour cela, on souhaite utiliser un objet, représentant le journal des évènements (*log*), sur lequel on pourra effectuer les actions suivantes :

- émettre un message d'information (*INFO*),
- émettre un message d'avertissement (*WARNING*),
- émettre un message d'erreur fatale (*FATAL*),

Chacune de ces actions sera réalisée par une méthode spécifique prenant en argument le message à émettre sous forme d'une chaîne de caractères.
On se figure que les applications clientes sont composées de nombreuses fonctions et/ou objets utilisant ou non le journal. On voudra donc à chaque utilisation interagir avec le même objet pour une application donnée.
Écrire une class **Log** offrant les fonctions citées et ayant les caractéristiques suivantes :

1. assurez vous que la classe **Log** ne puisse pas être dérivée,
2. assurez vous que la classe **Log** ne puisse pas être instanciée directement,
3. assurez vous que la classe **Log** puisse être instanciée en toute sécurité dans une application multithread,
4. on estime que le processus d'initialisation d'un journal est coûteux ; de plus on ne souhaite pas qu'un journal soit crée si l'application n'émet pas de messages (*i.e.* si elle n'utilise pas l'objet journal). Dans un commentaire placé avant le code de votre classe, vous rappellerez les deux types de singletons et justifierez votre choix dans ce cas,
5. fournissez un programme de test qui prouvera que l'objet de journalisation est bien unique ET que les caractéristiques ci-dessus (à l'exception de la protection multithread) sont bien réalisées. Pour cette question, vous placerez en commentaire le code empêchant la compilation de votre programme.

Note : pour faire simple, les messages émis seront simplement redirigés vers la sortie texte avec un préfixe permettant de les distinguer.

Travail à rendre : vous fournirez deux fichiers : celui de votre classe ainsi que celui de votre programme de test.

Voici un exemple d'utilisation où *XXX* représente du code caché pour éviter de donner des éléments de réponse :

```
class Test {
        int function1(int value) {
                XXX
                if (value>1000) XXX.warning("function1 a ete appelee avec une valeur el
                if (value==0)  XXX.fatal("function1 a ete appelee avec une valeur nulle
                        "Une division par zero va avoir lieu...");

                return 12345/value;
        }

        static void main(String args[]) {
                XXX
                XXX.info("Demarrage de l'application");
                ...
```

```
                XXX.info("Fin normale de l'application");
        }
}
```