Énoncé : **Système de gestion de configuration centralisée**

Contexte :

Vous travaillez sur un projet pour une entreprise fictive appelée **GlobalCorp**, qui développe une application avec plusieurs modules : gestion des utilisateurs, gestion des commandes et gestion des produits. Tous ces modules doivent accéder à une configuration centrale, qui contient des informations telles que :

- Les paramètres de connexion à la base de données.
- Les variables d'environnement (développement, production).
- Les clés d'API externes.

Afin d'assurer la **cohérence des paramètres** entre tous les modules, vous devez créer une classe qui permettra de centraliser cette configuration. Il est essentiel que cette classe :

1. **Soit partagée** entre tous les modules de l'application. Si un module modifie un paramètre de configuration, la mise à jour doit être immédiatement reflétée pour les autres modules.
2. **Ne puisse être instanciée qu'une seule fois** dans l'application. Il ne doit pas y avoir de multiples copies de la configuration, pour éviter les incohérences.
3. Contienne une méthode `getSetting(String key)` qui permet de récupérer la valeur d'un paramètre, et une méthode `setSetting(String key, String value)` pour modifier la valeur d'un paramètre.

Spécifications :

1. Vous devez créer une classe **Configuration** qui contiendra les paramètres globaux de l'application. Cette classe doit garantir qu'il n'y ait qu'une seule instance dans toute l'application.
2. La classe **Configuration** doit contenir un attribut `Map<String, String> settings`, qui stocke les différents paramètres.
3. Vous devez implémenter deux méthodes :
   - `getSetting(String key)` pour récupérer la valeur d'un paramètre.
   - `setSetting(String key, String value)` pour modifier ou ajouter un paramètre.
4. La première utilisation de la classe **Configuration** doit permettre de **charger les paramètres** à partir d'un fichier ou d'un ensemble par défaut (exemple : un fichier `config.properties`).
5. Toute tentative d'instancier une nouvelle **Configuration** doit renvoyer la même instance partagée.

Contraintes supplémentaires :

- Votre classe **Configuration** doit être **thread-safe**. Si plusieurs modules tentent d'accéder ou de modifier les paramètres simultanément, l'intégrité des données doit être préservée.
- L'application doit permettre de récupérer les paramètres de manière efficace, sans recréer l'objet ou recharger les paramètres depuis le fichier de configuration à chaque fois.