Exo1:

1. Quel est l'effet de static et de transient sur la sérialisation ?
   Champs qui poessede static et transient ne sont pas sérialisés dans une classe implémentant Serializable.

Exo3:

1.1 Que fait le code précédent ?
indiquant si l'hôte est joignable dans le délai spécifié (dans ce cas, 2 000 millisecondes ou 2 secondes).

1.2 la commande réseau équivalente au comportement de la méthode isReachable La commande ping : ping <adresse_IP> -> qu'est utilisée pour tester la connectivité réseau avec un hôte spécifique.

2. Que fait le code suivant ?
Ce code peut être utile pour vérifier la connectivité locale ou pour mesurer le temps de réponse d'un serveur

3. Quelle est la signification des deux autres paramètres : netif et ttl?

Le champ netif (NetworkInterface) : Permet de spécifier sur quelle interface réseau le test de connectivité doit être effectué.
Le champ TTL: indique la durée de vie maximale du paquet.

4. Quel est le scénario dans lequel on doit préciser le paramètre netif ? c'est-à-dire, quelle est la particularité de la machine à partir du quelle, nous avons lancé la méthode isReachable.

nous devrons préciser le paramètre netif lorsque on veut vérifier la connectivité via une interface réseau spécifique, comme dans les cas de machines multihomées, machines virtuelles ou machines avec des interfaces virtuelles:

1. Les machines multihomées : Une machine multihomée possède plusieurs interfaces réseau. Si vous souhaitez vérifier la connectivité via une interface spécifique, vous devrez préciser le paramètre netif.

2. Les machines virtuelles : Les machines virtuelles peuvent avoir plusieurs interfaces réseau, et vous devrez préciser le paramètre netif pour vérifier la connectivité via une interface spécifique.

3. Les machines avec des interfaces virtuelles (VLAN, VPN, etc.) : Les interfaces virtuelles peuvent être utilisées pour créer des réseaux virtuels, et vous devrez préciser le paramètre netif pour vérifier la connectivité via une interface spécifique.

Exo 4 :

1. Que fait le code suivant?

   Le code fourni lit le contenu d'un fichier "symptomes.txt", compte le nombre d'occurrences de chaque mot, et stocke les résultats dans une structure de données de type Map.

   La méthode traiter lit chaque ligne du fichier "symptomes.txt" et extrait les mots individuels en utilisant la classe StringTokenizer. Puis, elle appelle la méthode Ajouter pour chaque mot extrait, en passant le Map et le mot en question. La méthode Ajouter vérifie si le mot existe déjà dans le Map, et si oui, elle incrémente le nombre d'occurrences, sinon elle ajoute le mot avec une occurrence initiale de 1

2. la méthode compute() permet de mettre à jour ou d'ajouter une entrée dans une Map en utilisant une fonction lambda qui définit le comportement à appliquer sur la clé et la valeur correspondante

3. la classe **A** lit un fichier texte de symptômes, stocke chaque symptôme dans une **HashMap** et compte le nombre d'occurrences de chaque symptôme en utilisant la méthode **compute()** de la **Map**.

Exo 5 :

1. Exécuter le code précédent. Que fait cette classe ?

   Le code Java fourni lit le contenu du fichier "test1.txt", copie chaque caractère lu dans le fichier "test2.txt", puis ferme les flux de lecture et d'écriture.

4. Le code écrit les caractères ASCII de A à J dans le fichier "test1.txt". L'entier passé comme paramètre à la méthode **write()** représente le code ASCII du caractère à écrire.

Exo 6 :

1. Expliquer l'instruction flotEcriture.write("mirsdgl",0,8-i); C'est quoi la signification des 3 paramètres ?

cette instruction écrit une sous-chaîne de la chaîne "mirsdgl" dans un flux de sortie, en commençant à l'index 0 de la chaîne source et la valeur de i détermine la longueur de la sous-chaîne à écrire