Exercice 1 (12 pts) : choisir la bonne réponse :

1 point pour chaque réponse

1. L'instruction : ServerSocket ss=new ServerSocket('3000') ; signifie :
   - ☐ **Créer un server-socket sur le port 3000.**
   - ☐ Attendre les demandes de connexion sur le port 3000
   - ☐ Créer le socket coté serveur sur le port 3000.
   - ☐ Envoyer le message '3000' dans le socket.

2. L'instruction : Socket s= serverSocket.accept() ; permet de :
   - ☐ Créer un socket coté client.
   - ☐ Réserver un port de communication.
   - ☐ Accepter les messages arrivés dans le socket.
   - ☐ **Créer un socket coté serveur.**

3. Pour envoyer des données binaires et primitives (int, double, …) via un socket Java, le flux le plus approprié est : ?
   - ☐ DataInputStream
   - ☐ **DataOutputStream**
   - ☐ BufferedReader
   - ☐ PrintWriter

4. Lors de l'utilisation de `ServerSocket ss = new ServerSocket(8080)`, que se passe-t-il si le port 8080 est déjà utilisé par une autre application ?
   - ☐ Le serveur attend que le port se libère.
   - ☐ Le serveur choisit automatiquement le port suivant (8081).
   - ☐ **Une exception est levée.**
   - ☐ Le système d'exploitation partage le port entre les deux applications.

5. Dans une application Sockets en Java, l'utilisation des Thread permet principalement de :
   - ☐ Réduire le temps d'exécution du programme.
   - ☐ **Permettre au serveur de traiter plusieurs clients simultanément.**
   - ☐ Invoquer les méthodes à distances sans envoie de paramètres.
   - ☐ Éviter l'utilisation des sockets.

6. Dans une application Java RMI, une interface distante doit :
   - ☐ Implémenter Serializable.
   - ☐ extends UnicastRemoteObject.
   - ☐ **extends Remote.**
   - ☐ Être déclarée comme 'public static'.

7. Quelle méthode permet directement de lire une ligne envoyée via un socket ?
   - ☐ **readLine()**
   - ☐ recieveLine()
   - ☐ intputString()
   - ☐ Invocation directe de méthodes sans échange de données

8. Quelle est l'utilité du registre RMI (rmiregistry) ?
   - ☐ Stocker les fichiers .class du serveur.
   - ☐ Compiler le code source Java en code machine.
   - ☐ Gérer la base de données de l'application.
   - ☐ **Permettre aux clients de trouver les objets distants par leur nom.**

9. Quel port par défaut est utilisé par le rmiregistry de RMI?
   - ☐ **1099**
   - ☐ 8080
   - ☐ 80
   - ☐ 99

10. Dans une application Java RMI, comment le client se connecte-t-il à un serveur distant ?
    - ☐ En créant un socket avec new Socket(localhost, port)
    - ☐ En appelant la méthode getConnection()
    - ☐ En utilisant la méthode rebind()
    - ☐ **En invoquant la méthode lookup()**

11. Quelle est la fonction principale de CORBA ?
    - ☐ Gérer les requêtes HTTP dans les applications web modernes
    - ☐ **La communication entre objets distribués, indépendamment du langage.**
    - ☐ Faciliter l'interaction avec la base de données
    - ☐ Combiner les Socket et RMI dans la même application.

12. Comment Java RMI permet-il à un serveur de traiter plusieurs clients simultanément ?
    - ☐ On doit créer des threads pour chaque client.
    - ☐ Il est nécessaire de lancer plusieurs serveurs.
    - ☐ **RMI est multithreadé et gère automatiquement plusieurs clients.**
    - ☐ Chaque client doit obligatoirement se connecter via un port différent.

Exercice 2 (8 pts) : L'application suivante met en œuvre une communication client–serveur basée sur les sockets TCP en Java. *(le bloc try{} catch {} est supprimé pour faciliter la lecture du code).*

1- Déterminer le client et le serveur : le serveur est la classe **program2** , le client est la classe **program1**

Justifier : **program2 contient le serverSocket, la fonction accept(), contient la fonction de calcul … etc -> serveur**

1

……

……

……

……

……

```
1.  public class program1
2.  {
3.  int port = 2000 ;
4.  int  x=16;
5.  public program1 ()
6.  {
7.   Socket socket = new Socket("localhost", port);
8.   BufferedReader in =   new BufferedReader( new InputStreamReader(
     socket.getInputStream()));
9.   PrintWriter out= new PrintWriter (socket.getOutputStream(),true);
10.
11.  out.println(x);
12.  String rep=in.readLine() ;
13.  System.out.println("réponse du serveur = "+rep) ;
14. }
15. public static void main (String args[]) throws IOException
16. { new program1();}
}
```

```
1.  public class program2
2.  {
3.  int port = 2000 ;
4.  public String checkNumber(int n) {
5.    if (n <= 1) return "No";
6.    for (int i = 2; i * i <= n; i++)
a.     if (n % i == 0) return "No";
7.    return "Yes";
8.  }

9.  public program2(){
10.   Socket socket = serversoc.accept();
11.   ServerSocket serversoc = new ServerSocket(port) ;
12.   BufferedReader in = new BufferedReader(new
      InputStreamReader(socket.getInputStream()));
13.   PrintWriter out= new PrintWriter(socket.getOutputStream(),true);
14.   String x=in.readLine();
15.   out.println(checkNumber(Integer.parseInt(x)));
16. }
17. public static void main (String args[]) throws IOException
18. { new program2();}
19. }
```

2- Que fait la fonction checkNumber() ? **Vérifie si le nombre passé comme paramètre est premier ou non.** (1)

……

……

……

……

3- Le programme contient une erreur empêchant son exécution. Identifiez cette erreur : 1

Dans le programme serveur : la méthode accept() est placée avant la création du serverSocket ! (la ligne 10 et 11)

……

Proposez la correction appropriée ? 1

On inverse les ligne 10 et 11:

```
ServerSocket serversoc = new ServerSocket(port) ;
Socket socket = serversoc.accept();
```

……

……

4- En supposant que l'adresse IP du client soit 192.168.1.10 et celle du serveur 192.168.1.20, quelles modifications faut-il apporter au programme ?

Modifications dans le code client	Modifications dans le code serveur
Socket socket = new Socket("192.168.1.20", port); 1	**Aucune modification à faire** 1

5- Quelles modifications faut-il apporter au programme pour pouvoir gérer plusieurs clients simultanément (en 1 ou 2 phrases et sans coder) ?

Modifications dans le client	Modifications dans le serveur
Aucune modification à faire 1	**On utilise les threads** 1

Bon Courage.