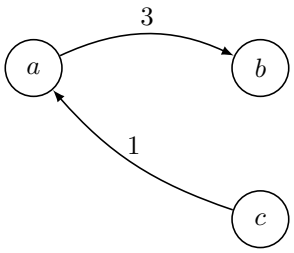
TP n° 4: Classe et héritage (deuxième partie)

Exercice 1 : Transmission des messages dans un graphe

On voudrait modéliser un graphe où chaque sommet envoie des messages à ses sommets successeurs, et en reçoit de la part de ses sommets prédécesseurs.
En utilisant les classes développées en TP3 et TP2, et en utilisant la classe `Thread` ou l'interface `Runnable`, écrire un programme Java qui permet les fonctionnalités suivantes :

1. Créer un graphe orienté et valué, où chaque sommet possède un seul numéro de ports et une adresse.
2. À la création du graphe, tous les sommets doivent être à l'écoute de leur port (utiliser l'API socket vu en TP3).
3. L'utilisateur doit créer un objet de la classe tp3.Message en précisant le sommet source, le sommet de destination et le message à envoyer.
4. En fonction de l'objet message créé, et en utilisant l'algorithme Dijkstra (vu en TP2), le sommet source correspondant doit envoyer cet objet au sommet de destination en suivant le plus court chemin.

Tester le programme sur un graphe simple comme celui de la figure ci-dessous, où le sommet `c` envoie un message au sommet `b` par l'intermédiaire de `a`



Exercice 2 : Transmission de plusieurs messages et File d'attente

Dans cet exercice il est demandé de modifier le programme de l'exercice 1 pour permettre d'envoyer plusieurs messages dans le graphe.

1. Chaque sommet possède une file d'attente FIFO qui contient les messages transmis dans le graphe
2. Chaque sommet doit pouvoir recevoir plusieurs messages provenant de plusieurs sommets prédécesseurs.
3. À la réception d'un message, le sommet doit le mettre dans la file d'attente
4. Toutes les **trois secondes**, le sommet doit prendre un message de la file d'attente et l'envoyer au sommet destinataire (selon le plus court chemin)
5. Si le sommet trouve la file vide, il doit attendre jusqu'à ce qu'elle contienne au moins un message
6. La file d'attente est une ressource partagée entre les deux Threads d'émission et de réception, elle ne doit dons pas être utilisée au même moment par ces deux processus.

Tester les programmes sur le graphe valué et orienté de l'exercice 2 de la fiche TP1.

Exercice 3 : Simulation

1. On voudrait regrouper toutes les classes développées jusque là dans une seule simulation ; où on doit créer un graphe orienté et valué.
2. Toutes les cinq secondes l'utilisateur est demandé de créer un message où il doit préciser le sommet source et le sommet de destination.
3. Utilisant l'algorithme de Dijkstra, les messages doivent être transmis d'un sommet à un autre selon le plus court chemin.
4. Les messages doivent être stockés dans les files d'attente de chaque sommet, celui-ci transmet les messages stockés a leur destination toutes les trois secondes.