Applications interactives
420-3GP-BB
Travail pratique #1
Automne 2022
Version 1.0

Directives :

- Le travail est individuel. Il vaut pour 30% de la note finale de la session.
- Le travail doit être remis par GitHub avant la date de remise indiquée sur Lea
- Vous devez vous créer un dépôt GitHub pour ce projet avec ce lien https://classroom.github.com/a/gI3JfQ_D
- Renommez ensuite votre dépôt de la façon suivante :
  a22-tp1-<votre DA> par exemple a22-tp1-1234567

Description du travail :

Vous devez programmer une simulation d'un système de répartition de voyages pour une compagnie de taxis. Les taxis circuleront dans une zone afin d'amener un client d'un point à l'autre dans la même zone. L'objectif de ce travail est de pratiquer les threads. Vous aurez donc à implémenter le tout avec des threads.

La simulation se fera dans une zone de 50 unités par 50 unités. Deux téléphonistes recevront des demandes de client afin d'effectuer un voyage entre deux points à l'intérieur de la zone. Un répartiteur assignera un voyage à un taxi qui se dirigera immédiatement vers le point de départ du voyage. Une fois au point de départ du voyage, le taxi se dirigera vers le point d'arrivée, ce qui terminera le voyage.

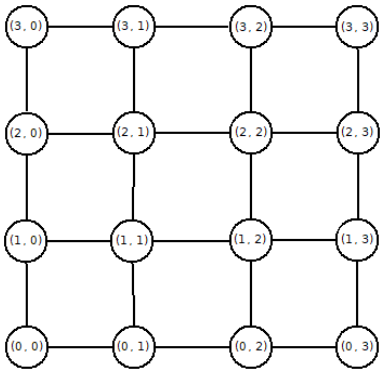
Le programme se terminera lorsque tous les voyages seront terminés.

Les principaux threads requis seront les suivants :

- Un thread pour chaque téléphoniste
- Un thread pour le répartiteur
- Un thread pour chaque taxi

Zone de service

Les taxis se déplacent dans une zone de 50 unités par 50 unités. Les coordonnées de cette zone iront de (0, 0) à (49, 49). Les voyages se feront également dans cette zone. À titre d'exemple. voici une zone analogue mais de 4 unités par 4 unités :



Vous ne devez pas dessiner cette zone à l'écran. Vous n'avez même pas besoin de la représenter en mémoire. Toutefois, les taxis et les voyages utiliseront ce système de coordonnées. On doit voir les coordonnées comme des coins de rue et les lignes comme des rues. Par exemple, si un taxi situé à la position (0, 1) doit aller chercher un client à la position (2, 3), il pourra passer de (0, 1) à (1, 1), de (1, 1) à (2, 1), de (2,1) à (2,2) et finalement de (2, 2) à (2, 3).

Les voyages

Un voyage comprendra simplement deux coordonnées dans la zone de service. Un point de départ et un point d'arrivée. Ces deux coordonnées devront être déterminées de façon aléatoire. Les deux coordonnées doivent être différentes.

Pour qu'un voyage soit complété, il faut que le taxi à qui le voyage a été assigné vienne à la position de départ dans le but de prendre le client et ensuite se déplace à la position d'arrivée pour déposer le client à sa destination. Les voyages doivent être numérotés (à partir de 1).

Les taxis

Lors de leur création, les taxis devront se faire allouer une position de départ initiale dans la zone de service. Un taxi sera soit en attente ou encore en mouvement. Un taxi en attente reste tout simplement en place. Il peut toutefois se faire assigner un voyage. Dans ce cas, il se déplacera graduellement vers la position de départ du voyage et ensuite vers la position d'arrivée du voyage. Lorsque le voyage est terminé, le taxi reste en place et attend sa prochaine assignation. Les taxis devront être numérotés (à partir de 1). Il y aura 5 taxis.

Threads des téléphonistes

Il y aura deux téléphonistes. Les téléphonistes ont pour mission de prendre des appels et de créer des voyages. Tel que précédemment mentionné, un voyage prend deux

coordonnées différentes générées aléatoirement. Une fois qu'un voyage est créé, le téléphoniste place le voyage dans une structure afin que le répartiteur puisse l'assigner à un taxi en attente.

Chaque téléphoniste créera 20 voyages[1]. Chaque voyage doit être créé à un intervalle de 1000 à 2000 millisecondes. La valeur doit être générée aléatoirement à chaque création de voyage.

Thread du répartiteur

Le répartiteur aura comme mission d'assigner les voyages créés par les téléphonistes à des taxis. Pour qu'un voyage soit assigné à un taxi, ce dernier doit être en attente. On ne peut jamais assigner un voyage à un taxi qui est en mouvement.

Lorsqu'il assigne un voyage à un taxi, le répartiteur doit sélectionner le voyage le plus près de la position actuelle du taxi. Lorsque le voyage est assigné à un taxi, il ne doit pas être réassigné à un autre taxi ensuite.

Threads des taxis

Il y aura un thread pour chaque taxi. Lorsqu'un taxi est en attente, il ne fait rien. Lorsqu'un voyage lui est assigné, il se déplace dans un premier temps vers le point de départ du voyage. Une fois rendu au point de départ, il se déplace au point d'arrivée.

Une fois rendu au point d'arrivée, il retombe en attente.

Les déplacements se font d'un coin de rue à un autre. Chaque déplacement entre deux coins de rue prend entre 250 et 500 millisecondes (généré aléatoirement à chaque déplacement).

Au démarrage du programme, chaque taxi est placé à un endroit aléatoire dans la zone de service.

Affichage de la trace d'exécution

Vous devrez afficher dans la console toutes les informations requises (voir la trace d'exécution). Vous n'êtes pas tenu de respecter intégralement le format d'affichage mais votre trace doit être facile à suivre. Vous devrez afficher :

- Le numéro des taxis et leur position initiale
- Le numéro des voyages et les coordonnées de départ et d'arrivée.
- L'assignation des voyages à un taxi. Indiquez le numéro de taxi et de voyage
- Quand un taxi arrive à la position de départ et à la position d'arrivée d'un voyage, affichez un message qui indique ce qui se passe, le numéro de voyage, le numéro de taxi et la position.

1 Pour un total de 40 voyages

Choix de la structure de données pour les voyages

Les voyages doivent être placés dans une structure de données. Vous devez choisir une structure appropriée pour les exigences du logiciel. ATTENTION : Vous pourriez avoir à synchroniser les accès à la structure avec des verrous.

Contraintes :

- Le travail doit être réalisé en C# avec .NET 6.0. Le projet doit pouvoir s'exécuter sur Visual Studio 2022.
- Votre programme doit utiliser les threads.
- Vous devez utiliser une approche orientée-objet. Vos classes doivent être encapsulés correctement.

Barème de correction

Qualité de la programmation		**20 %**
▪ *Commentaires*	*5 %*	
▪ *Découpage fonctionnel*	*5 %*	
▪ *Optimisation*	*5 %*	
▪ *Clarté générale du code*	*5 %*	
Fonctionnement		**80 %**
(Un programme qui ne compile pas peut se mériter 0 dans cette partie)		
▪ *Threads des téléphonistes*	*5 %*	
▪ *Thread du répartiteur*	*5 %*	
▪ *Threads des taxis*	*5 %*	
▪ *Utilisation judicieuse des verrous*	*10 %*	
▪ *Création des voyages*	*5 %*	
▪ *Assignation des voyages aux taxis*	*15 %*	
▪ *Déroulement correct des voyages*	*15 %*	
▪ *Choix de la structure de données*	*5 %*	
▪ *Affichage de la trace*	*10 %*	
▪ *Sortie du programme sans erreur*	*5*	*%*

La qualité de la programmation et le fonctionnement sont deux critères différents. Toutefois, pour courir la chance d'avoir tous vos points pour la qualité de la programmation, vous devez tenter d'avoir fait l'ensemble du travail. Si des parties du travail sont absentes, vous serez pénalisé dans la qualité de la programmation en proportion de la partie du travail qui n'a pas été faite.

Le non-respect des contraintes entraînera une sévère pénalité non-prévue à cette grille.

Annexe

La trace est assez volumineuse. Voici toutefois deux extraits qui pourront vous donner une idée :

Au démarrage

```
Le taxi 1 est en fonction à (33, 19)
Le taxi 2 est en fonction à (30, 9)
Le taxi 3 est en fonction à (22, 44)
Le taxi 4 est en fonction à (31, 7)
Le taxi 5 est en fonction à (4, 10)
Creation voyage 1 Depart: (25, 2) Arrivee:(8, 27)
Voyage 1 assigné au taxi 1
Creation voyage 2 Depart: (31, 9) Arrivee:(26, 4)
Voyage 2 assigné au taxi 2
Client du voyage 2 entre dans taxi 2 à (31, 9)
Creation voyage 3 Depart: (43, 41) Arrivee:(45, 7)
Voyage 3 assigné au taxi 3
Creation voyage 4 Depart: (14, 40) Arrivee:(44, 30)
Voyage 4 assigné au taxi 4
Creation voyage 5 Depart: (16, 20) Arrivee:(3, 35)
Voyage 5 assigné au taxi 5
Creation voyage 6 Depart: (10, 5) Arrivee:(22, 26)
Client du voyage 2 sort du taxi 2 à (26, 4)
Voyage 6 assigné au taxi 2
Creation voyage 7 Depart: (0, 32) Arrivee:(12, 10)
Creation voyage 8 Depart: (2, 47) Arrivee:(45, 38)
Creation voyage 9 Depart: (20, 33) Arrivee:(45, 5)
Creation voyage 10 Depart: (31, 29) Arrivee:(11, 28)
Creation voyage 11 Depart: (40, 24) Arrivee:(13, 28)
Creation voyage 12 Depart: (47, 38) Arrivee:(3, 16)
```

À la fin :

```
Client du voyage 21 sort du taxi 2 à (11, 1)
Voyage 34 assigné au taxi 2
Client du voyage 14 sort du taxi 5 à (30, 29)
Voyage 18 assigné au taxi 5
Client du voyage 34 entre dans taxi 2 à (5, 5)
Client du voyage 34 sort du taxi 2 à (9, 7)
Client du voyage 8 entre dans taxi 1 à (2, 47)
Client du voyage 19 entre dans taxi 4 à (6, 5)
Client du voyage 19 sort du taxi 4 à (1, 12)
Client du voyage 12 sort du taxi 3 à (3, 16)
Client du voyage 18 entre dans taxi 5 à (2, 6)
Client du voyage 18 sort du taxi 5 à (11, 7)
Client du voyage 8 sort du taxi 1 à (45, 38)

Sortie de C:\Users\Eric Wenaas\source\repos\420-3GP-BB-Travaux\TP1\ThreadTaxi\bin\Debug\net6.0\ThreadTaxi.exe (processus 7748). Code : 0.
Pour fermer automatiquement la console quand le débogage s'arrête, activez Outils->Options->Débogage->Fermer automatiquement la console à l'arrêt du débogage.
Appuyez sur une touche pour fermer cette fenêtre. . .
```