TD 1 : Modélisation avec diagramme de classe et implémentation en Java

Problématique : Ma banque est-elle efficace ?

Vous venez d'être nommé directeur d'une nouvelle agence de banque. Par une étude du marché, on suppose que le nombre moyen des clients à traiter chaque jour dans votre agence reste stable. Votre première mission consiste à déterminer le nombre d'employés (caissiers) à embaucher pour rassurer le bon fonctionnement de l'agence.

Pour ce faire, il y a principalement deux aspects à prendre en compte :

1) Les caissiers ne doivent être ni trop libres ni trop chargés. Quand ils sont trop libres, cela signifie que l'on en embauche plus que nécessaire. Quand ils sont trop chargés, cela signifie qu'il n'y en a pas assez et qu'il y a une surcharge de travail.
2) Pourtant, le taux d'occupation des caissiers ne permet pas de déduire le temps moyen d'attente des clients, qui ne doit pas dépasser un certain seuil en raison de satisfaction. Ceci fait le deuxième aspect important d'un point de vue des statistiques.

Vous avez ainsi besoin de concevoir un simulateur qui vous permet de simuler une période de déroulement de l'agence bancaire (un jour, une semaine…) et d'avoir des résultats statistiques sur les aspects mentionnés ci-dessus, afin de prendre des décisions (embaucher plus ou moins de cassiers).

Dans un premier temps, on modélise uniquement les éléments les plus fondamentaux de la simulation. Cependant, la conception doit rester extensible afin de pouvoir y rajouter d'autres aspects ultérieurement.

Caractéristiques principales de la simulation :

La simulation se déroule pendant un certain nombre de temps unitaires. Attention, il ne s'agit pas de simulation en temps réel (le système n'a pas besoin de « s'endormir » entre unités de temps pour simuler le temps réel), parce que seuls les résultats statistiques calculés à l'issue de la simulation nous intéressent. Par exemple, imaginons tout simplement que chaque unité de temps correspond à une ou plusieurs minutes dans la journée. La durée totale de la simulation doit être paramétrable.

Ainsi, l'horloge de la simulation se présente en unité de temps. Au début, unité 0, et puis 1, 2, 3, 4, 5… On suppose que pendant la simulation, les clients arrivent aux moments « uniformément répartis ». C'est-à-dire qu'un client arrive à tous les N (paramétrable) unités de temps. Par exemple, si N = 5, un nouveau client arrive à 0, 5, 10, 15, 20 … unité de temps.

Chaque caissier met un temps aléatoire (entre deux valeurs min et max paramétrables) pour traiter un client. Quand un client arrive, si aucun cassier n'est libre, on le met dans la file d'attente, sinon, le premier caissier libre trouvé va le servir. Il n'y a qu'une seule file d'attente partagée par tous les caissiers. Le nombre de caissiers doit être certainement paramétrable afin d'analyser des résultats de différentes simulations avec différentes entrées.

Il s'agit donc, pour chaque itération de la simulation, de mettre à jour l'état des caissiers et de la file d'attente, et de traiter un client arrivant le cas échéant. Pour simplifier, la fin de la simulation s'arrête quand la durée prédéfinie est atteinte, même si certains cassiers ne finissent pas encore le service du client courant.

A faire :

1) Proposez une solution à cette problématique par un diagramme de classe UML. Suivez les trois étapes présentées en cours.

2) Reprenez la solution proposée par l'enseignant : étudiez le diagramme de classe et le programme* fourni. Complétez les méthodes manquantes dans le programme.

* Pour utiliser le programme fourni par l'enseignant, il suffit de copier le code (avec les packages) dans « src » d'un « Java Project » sous Eclipse.