Le buffet des ogres 0.1

Département d'informatique
Cégep de Jonquière

Programmation avancée – Laboratoire 1 – Entity, programmation concurrente – 10%

Consignes

- Le travail est à faire en équipe de deux ou seul.
- Le laboratoire est constitué de deux solutions (`Ogres` et `Cuisine`). Créez deux dépôts GitHub (`OgresNomPrenom` et un pour `CuisineNonPrenom`) et partagez-les avec votre coéquipier et avec reblapointe@gmail.com
- Remplissez la grille d'évaluation individuelle pour vous et votre coéquipier sous forme de formulaire Teams : https://forms.office.com/r/u9aG0iQZmB

Mise en situation

Nous allons programmer un simulateur de buffet dans lequel un cuisinier sert des plats sur une table et trois (`NB_OGRES`) ogres mangent les plats.

Le cuisinier prend en moyenne 2 secondes (`TEMPS_MOYEN_PREPARATION_PLAT`) pour préparer un plat. Un plat a une taille et une date de création. Un plat a une taille moyenne de 5 bouchées (`TAILLE_PLAT_MOYEN`).

Le temps de préparation et la taille des plats suit une distribution exponentielle, de minimum une bouchée. Vous pouvez simuler ainsi :

```
Random rnd = new Random();
valeur = (int)(-1 * (MOYENNE – 1) * Math.Log(1 - rnd.NextDouble())) + 1;
```

Quand un plat est prêt, il est mis sur une table de buffet. Les ogres peuvent prendre le plat de leur choix. Chaque ogre mange un plat à la fois. Il prend une bouchée par seconde. Quand il a terminé, il prend un autre plat. Si la table de buffet est vide, l'ogre attend.

Un plat ne peut pas être mangé par deux ogres en même temps.

Contraintes

A) Vous allez créer deux solutions, contenant chacune un projet console C# .NET Core : `Cuisine` et `Ogres`.

   Le projet `Cuisine` produit les plats. À chaque fois qu'un plat est généré, il est affiché à la console.

   Le projet `Ogres` consomme les plats. À chaque fois qu'un ogre sélectionne un plat, la console est rafraichie. Celle-ci doit afficher les ogres, les plats qu'ils sont en train de manger et le contenu de la table.

   Le contenu des consoles doit être lisible et intuitif.

B) Les plats doivent être sauvegardés dans une base de données générée par le code en utilisant *Entity*.

C) Votre exécution doit être asynchrone ou concurrente, avec un fil par ogre. Le code doit être *thread-safe*.

D) Les paramètres de la simulation doivent être dans des constantes. Votre code doit fonctionner pour toute valeur des paramètres :

   Dans `Cuisine` :
   - `int TEMPS_MOYEN_PREPARATION_PLAT` (en millisecondes)
   - `int TAILLE_PLAT_MOYEN` (en nombre de bouchées)

   Dans `Ogres` :
   - `int NB_OGRES`

Fonctionnalités facultatives

Ces fonctionnalités ne sont pas obligatoires, mais rendent votre application plus réaliste et flexible.

A) Donnez une capacité maximale à la table. (Par exemple : `CAPACITE_TABLE` = 100 plats).

B) Terminez la simulation après une durée prédéterminée (`DUREE_SIMULATION`). Une fois la simulation terminée, donnez des statistiques de la simulation.

Grille de correction (30 points)

Critère	Excellent (≈100%)	Satisfaisant (≈80%)	Acceptable (≈60%)	Peu satisfaisant (≈40%)	Insatisfaisant (≤20%)
Fonctionnement général de l'application /13	- L'application réalise toutes les fonctionnalités et les surpasse - L'application ne plante jamais	- L'application réalise toutes les fonctionnalités - L'application ne plante pas dans un usage normal	- L'application réalise la plupart des fonctionnalités - L'application plante dans un usage limite	- L'application réalise les fonctionnalités essentielles- L'application plante, même dans un usage parfait	- L'application ne réalise pas les fonctionnalités -L'application ne roule pas ou ne compile pas
Qualité de la conception et de la programmation /8	- Les classes respectent impeccablement le concepts oo - Gestion parfaite des exceptions - Cohésion maximale et couplage minimal - Algorithmes optimaux - Aucun mauvaise pratique de programmation - Les conventions de programmation sont toutes respectées, sans exception	- Les classes respectent le concepts oo - Gestion adéquate des exceptions - Cohésion très forte et couplage très faible - Algorithmes efficaces - Aucun mauvaise pratique grave de programmation. Peu de mauvaises pratiques modérées. - Les conventions de programmation sont toutes respectées, avec quelques oublis rares	- Les concepts oo sont majoritairement respectés - Gestion acceptable des exceptions - Cohésion forte et couplage faible - Algorithmes acceptables - Peu de mauvaise pratique de programmation - La plupart des conventions de programmation sont respectées	- Les concepts oo sont souvent respecté - Gestion des exceptions peu satisfaisante - Cohésion faible et couplage inutile - Algorithmes lents - Plusieurs mauvaises pratiques de programmation - Plusieurs conventions de programmation ne sont pas respectées	- Les concepts oo ne sont pas respecté - Gestion des exceptions peu manquante - Code spaghetti - Algorithmes très lents - Code contenant beaucoup de mauvaises pratiques de programmation - Le non-respect des conventions rend le code illisible
Qualité de l'intégration des nouveaux concepts de programmation /6	- Utilisation optimale des fils d'exécution - Utilisation optimale d'Entity	- Utilisation adéquate des fils d'exécution - Utilisation adéquate d'Entity	- Utilisation acceptable des fils d'exécution - Utilisation acceptable d'Entity	- Utilisation des fils d'exécution peu satisfaisante - Utilisation d'Entity peu satisfaisante	- Utilisation des fils d'exécution manquante - Utilisation d'Entity peu manquante
Qualité du processus de développement /3	- Développement autonome et rigoureux - Respect parfait de la structure demandée - Persévérance soutenue dans le développement et dans la tenue du Git	- Développement principalement autonome et rigoureux - Respect de la structure demandée - Persévérance dans le développement et dans la tenue du Git	- Développement avec aide, assez rigoureux - Respect majoritaire de la structure demandée - Persévérance minimale dans le développement et dans la tenue du Git	- Développement avec beaucoup d'aide, peu rigoureux - Respect de la structure demandée en partie - Manque de persévérance dans le développement et dans la tenue du Git	- Développement non autonome, non rigoureux - Non-respect de la structure demandée - Pas de persévérance. Non tenue du Git