Université du Maine

TP 2 POO (Ruby)

Etude d'une chaîne d'usinage

Les réponses aux questions doivent figurer dans le listing sous forme de commentaires
Vous devez générer une documentation au format RDOC.

Pour mesurer les performances d'une chaîne d'usinage, il est moins coûteux de la simuler par programme que de la construire réellement. Le but du problème est donc de simuler le fonctionnement d'une telle chaîne.

1-Fonctionnement d'une chaîne

Une chaîne d'usinage est constituée d'une suite de postes de travail terminée par exactement un poste de sortie. Chaque poste réalise une partie du travail à effectuer sur les pièces. Pour être complètement traitée, une pièce passe successivement par chacun des postes de travail de la chaîne (du premier au dernier) puis est déposée sur le poste de sortie.

Chaque poste de travail traite au plus une pièce a la fois et les postes peuvent travailler en parallèle : si la chaîne comporte N postes de travail, alors on pourra avoir jusqu'a N pièces en cours de traitement. Tant qu'un poste de travail est occupe avec une pièce, il ne pourra pas commencer a en traiter une nouvelle ; par ailleurs, les pièces arrivent sur le poste de sortie dans l'ordre dans lequel elles sont entrées dans la chaîne : une pièce ne peut en doubler une autre.

Le temps de traitement d'une pièce par un poste de travail dépend soit du poste (ce temps alors le même pour toutes les pièces traitées par ce poste), soit de la pièce elle-même (c'est la pièce qui détermine le temps de traitement et ce temps est alors le même quelque soit le poste). Quand une pièce arrive sur un poste de travail, elle attend d'abord que les pièces arrivées avant elle aient été traitées. Cette attente, éventuelle, ne bloque pas le poste précèdent qui peut continuer a traiter des pièces[1].

2 - Mesures à faire

Pour une chaîne d'usinage, vide de pièces, et un certain nombre d'entrées datées de pièces dans cette chaîne, le responsable de l'usine souhaite connaître les informations suivantes :

1. Informations par poste de travail :
   - le nombre de pièces qu'il a traitées.
   - les durées d'activité et d'inactivité du poste[2] entre la date de début de traitement de la première pièce et celle de fin de traitement de la dernière pièce. Ces durées seront nulles si aucune pièce n'a été traitée.
2. Informations connues du poste de sortie :
   - le nombre de pièces complètement traitées,
   - et si ce nombre n'est pas nul :
     - le maximum des durées d'attentes des pièces ayant traversé cette chaîne,
     - la moyenne des attentes.

[1] On considérera, pour simplifier, que le nombre de pièces en attente d'un poste de travail n'est pas borne ; autrement dit, un poste lent ne peut pas empêcher ses prédécesseurs plus rapides de travailler.

[2] Un poste est actif quand il traite une pièce.

3 - Réalisation

Les dates et les durées seront exprimées en nombres entiers d'unités de temps[3]. On appellera mouvement le fait qu'a une certaine date, une pièce se présente a un poste (cette date ne correspond pas forcement au début du traitement de la pièce par le poste, car il peut y avoir déjà d'autres pièces en attente de traitement pour ce poste).

Evidemment, pour que la simulation donne des résultats corrects, il faut que le programme joue les mouvements dans leur ordre chronologique. a un instant donne, plusieurs pièces peuvent se trouver dans la même chaîne : il faudra jouer le mouvement le plus ancien. On utilisera un échéancier unique pour gérer les mouvements qui n'ont pas encore été joués. Auprès de cet échéancier, on pourra obtenir, en l'en supprimant, le mouvement le plus ancien (c'est a dire le prochain mouvement a jouer), et on pourra y ajouter de nouveaux mouvements.

Nous avons alors les classes suivantes : `Piece`, `Poste`, `Mouvement` et `Echeancier`.

3.1 Les méthodes de `Pièce`

- `attente()` : Méthode publique qui renvoie la durée totale d'attente de la pièce (entier).
- `attendre(duree)` : Méthode publique, qui augmente la durée totale d'attente de `duree`.
- `dureeTraitement()` : Méthode publique qui renvoie la durée de traitement de la pièce (entier), utile dans le cas d'un poste dont la durée de traitement dépend de la pièce traitée. La valeur de `dureeTraitement()` sera fixée a la construction de la pièce.

Q 1. Ecrire la classe `Piece`.

3.2 Les méthodes de `Mouvement`

- `date()` : Méthode publique qui renvoie la date a laquelle ce mouvement doit être réalisé (entier).
- `piece()` : Méthode publique qui renvoie la pièce concernée par ce mouvement.
- `poste()` : Méthode publique qui renvoie le poste qui doit traiter la pièce

Q 2. Ecrire la classe `Mouvement`.

3.3 Les méthodes de `Echeancier`

- `vide()` : Méthode publique qui renvoie vrai si cet échéancier ne contient aucun mouvement.
- `prochain()` : Méthode publique qui renvoie le plus ancien mouvement et le supprime de cet échéancier. Un message d'erreur est envoyé s'il est vide.
- `enregistrer(mouvement)` : Ajoute un nouveau mouvement à cet échéancier.

Q 3. Ecrire la classe `Echeancier`

4 - Les postes

Les méthodes de la classe `Poste` sont :

- `traiter(date, piece)` : Méthode publique qui correspond a demander au poste de traiter la pièce a l'instant date.
- `printStat()` : Méthode publique qui imprime les informations demandées par le responsable de l'usine

[3] L'unité de temps n'est pas précisée.

Cas des postes de travail :

- La méthode `printStat()` imprime les informations de ce poste de travail, puis appelle la méthode `printStat()` de son successeur.
- La méthode `traiter()` doit mettre a jour les informations du poste, mais aussi, et surtout, calculer a quelle date le traitement de pièce sera terminé :
  - si le poste est libre, il peut facilement calculer à quelle date le traitement sera terminé.
  - s'il est déjà occupé, ce n'est pas beaucoup plus compliqué, à condition que le poste connaisse la date à laquelle il aura terminé de traiter la pièce précédente.

Une fois la date de fin de traitement connue, il reste à la méthode `traiter()` à enregistrer le prochain mouvement de la pièce auprès de l'échéancier. Pour faire tout cela, le poste de travail doit connaître le poste qui le suit dans la chaîne ainsi que l'échéancier de la simulation.

Comme on l'a dit au début, il y a deux sortes de postes de travail qui ne différent que par la manière de calculer la durée de traitement d'une pièce. Les postes constants ont une durée de traitement constante et fixée lors de leurs créations. Les postes variables ont une durée de traitement égale à la durée de traitement de la pièce considérée.

Cas des postes de sortie :

- La méthode `printStat()` imprime les informations demandées.
- La méthode `traiter()` met a jour les informations.

Un exemple : Soit la classe `ChaineUsinage` suivante :

```
class ChaineUsinage {
    def ChaineUsinage.test()
        e = Echeancier.new();
        sortie = PosteSortie.new();
        p3 = PosteConstant.creer(e, sortie, 10);
        p2 = PosteVariable.creer(e, p3);
        p1 = PosteConstant.creer(e, p2, 6);
        e.enregistrer(Mouvement.creer(20, Piece.creer(3), p1));
        e.enregistrer(Mouvement.creer(21, Piece.creer(7), p1));
        e.enregistrer(Mouvement.creer(22, Piece.creer(5), p1));
        while (!e.vide()) {
            m = e.prochain();
            m.poste().traiter(m.date(), m.piece());
        }
        p1.printStat();
    end
}
```

L'exécution du programme suivant :

```
ChaineUsinage.test()
```

Pourra imprimer :

```
Travail (Constant-3 (6)) : Nb pièces traitées = 3, Activité = 18, Inactivité = 0
Travail (Variable-2) : Nb pièces traitées = 3, Activité = 15, Inactivité = 3
Travail (Constant-1 (10)) : Nb pièces traitées = 3, Activité = 30, Inactivité = 0
Sortie : 3 pièces traitées ; Attente max = 16, Moyenne = 7
```

Q4. Dessiner la chaîne construite par l'exemple.

Q5. Expliquer pourquoi le poste de travail variable a une durée d'inactivité de 3.

Q6. Ecrire les différentes classes qui implémentent les postes (de travail et de sortie) en factorisant si nécessaire.