Université Paris Cité

# Compléments en Programmation Orientée Objet
# Projet de session 2 : Gestion et simulation de projet

*Projet à réaliser seul, à terminer avant le 24 juin.*

## 1 Avant-propos

Le projet consiste en l'implémentation d'un logiciel de gestion et de simulation de projet avec allocation de ressources.
Nous insistons ici sur la conception objet (abstractions objet pertinentes et patrons de conception). Il ne suffira pas de rendre un programme qui fonctionne pour avoir une bonne note ! Par ailleurs, attendez-vous à des questions concernant vos choix techniques et soyez prêts à les défendre à l'oral.

## 2 Instructions de rendu

- Vous travaillerez sur un dépôt git privé hébergé sur le serveur `https://moule.informatique.univ-paris-diderot.fr` et veillerez à faire des commits réguliers.
- Vous donnerez l'accès à votre dépôt (notamment à son historique, dans le doute choisissez *maintainer*) aux enseignants du module.
- Vous déposerez sur moodle un lien vers votre git (dans un fichier texte). [1].
- Le point d'entrée de votre documentation sera le fichier `README.md` situé à la racine de votre dépôt dans la branche par défaut.
- Inscrivez-vous pour votre soutenance (salle réservée pour les 26, 27 et 28 juin) via Moodle [2].
- Les soutenances ont lieu en salle 2031 (Sophie Germain), faites en sorte de pouvoir exécuter votre projet là-bas (PC portable ou bien PC fixe de la salle de TP).

## 3 Les projets

Ici, un projet consiste en un ensemble de tâches, où chaque tâche est caractérisée par :
- son nom
- sa durée
- ses dépendances (d'autres tâches)
- les ressources requises

Ce cadre est très générique et peut représenter de nombreuses situations réelles. Par exemple :
- un cursus universitaire : les tâches sont les UE, leurs dépendances sont les prérequis pédagogiques, leurs ressources sont les heures d'enseignement (bien sûr en nombre limité dans l'emploi du temps hebdomadaire des étudiants du cursus)
- un projet de développement logiciel : les tâches sont les items de backlog, les ressources sont notamment les développeurs.

## 4 Votre programme

Votre programme doit permettre de définir un tel projet, un environnement d'exécution (la donnée des ressources disponibles) et de lancer l'exécution du projet en temps "réel" (avec un facteur d'échelle paramétrable, pour qu'on puisse voir les tâches s'exécuter en un temps raisonnable sous nos yeux... pensez à la soutenance !).

1. `https://moodle.u-paris.fr/mod/assign/view.php?id=1145254`
2. `https://moodle.u-paris.fr/mod/scheduler/view.php?id=1145257`

L'exécution du projet consiste simplement à faire des affichages pour chaque tâche qui démarre ou se termine, depuis des *threads* différents (plus de détails donnés après).
Vous pouvez envisager la progression suivante :

1. Les tâches n'utilisent pas de ressources (il faut juste vérifier les dépendances).
2. Les tâches utilisent chacune une ressource ; toutes les ressources sont interchangeables (pas de "types"), mais il en existe un certain nombre total fixé.
3. Au choix :
   (a) les tâches utilisent plusieurs ressources (sans types) ;
   (b) ou les tâches utilisent chacune une ressource typée (les ressources disponibles ont chacune un type fixé).
4. Les tâches utilisent un certain nombre de ressources de plusieurs types (p. ex. : cette UE a besoin de trois enseignants et de deux salles de TP et d'un amphi).
5. Pareil, mais les tâches ont des types et des sous-types (p. ex. : cette UE a besoin d'un professeur et de trois enseignants quelconques).

## 5 Contraintes imposées

**Architecture :** les classes servant à la description d'un projet (définissant une structure d'objets non modifiables) et les classes de gestion de l'exécution du projet (qui contiennent l'état d'exécution et les éventuels verrous) sont dans des *packages* séparés, le premier ne dépendant pas du second.

**Exécution parallèle :** les tâches doivent l'exécuter en parallèle autant que possible. Notamment, toute tâche dont les dépendances sont satisfaites et pour laquelle les ressources sont disponibles doit être exécutée immédiatement [3] en parallèle de toute autre tâche déjà en exécution (sur un autre *thread*).
Il est indispensable de garantir la terminaison du projet en temps borné (notamment il faut s'assurer qu'il n'y a pas de *deadlock*).

**Démonstration/cas de test :** pour la fiabilité de votre code, mais aussi pour pouvoir faire une soutenance concluante, prévoyez des cas d'exécution non triviaux démontrant que votre programme fonctionne pour tous les cas prévus :
- des convergences : des tâches qui dépendent effectivement de plus de une autre tâche
- des divergences : plusieurs tâches qui ont le même ensemble de dépendances
- des conflits de ressources (surtout intéressant pour des tâches qui sont disponibles en même temps, c'est-à-dire qui ont toutes leurs dépendances satisfaites en même temps)

**Observabilité :** Les exécutions doivent être visibles. Notamment, faites un affichage au début et à la fin de chaque tâche. Chaque affichage doit donner le nom de la tâche, le temps courant ainsi que le nom du *thread* qui fait cet affichage. Le début de tâche doit indiquer en plus l'identité des ressources utilisées.

## 6 Hors sujet

On ne demande pas d'interface graphique. On ne demande pas non plus de programmer un algorithme d'optimisation du temps d'exécution. Tant que l'exécution est parallélisée et qu'elle est correcte (toutes les tâches sont exécutées, et ce en respectant leurs contraintes), cela suffit.

3. À moins que cela permette une diminution de la durée totale d'exécution du projet, mais dans ce cas il faut le justifier.

# 7 Conseils

**Concurrence :** le plus simple c'est de confier chaque tâche à un thread virtuel, qui attend d'abord que les dépendances soient satisfaites, puis attend que les ressources soient disponibles et les réserve dans la foulée le cas échéant, exécute le contenue de la tâche, libère les ressources, change le status de la tâche à "terminée" (et en avertit les autres tâches, éventuellement).
Il est aussi possible d'utiliser `ForkJoinTask`/`ForkJoinPool`.
(Posez-vous la question de la pertinence des *virtual threads* et de `ForkJoin` par rapport aux autres techniques vues en cours.)

**Modélisation :** pour les données du projet, pensez aux `enum`, aux `record`, etc.
Pensez aux patrons de conception vus en cours pour la construction du projet à exécuter (plusieurs peuvent s'appliquer). Cela pourrait faciliter l'écriture d'exemples multiples.

**Autres questions ?** N'hésitez pas à venir sur le forum dédié : `https://moodle.u-paris.fr/mod/forum/view.php?id=1145251`.

# 8 Critères d'évaluation

**Le rendu :**

- Le programme est écrit en Java (si version différente de 21, précisez le dans votre `README.md`).
- Le `README.md` indique comment compiler, exécuter et utiliser votre programme et ses tests.
- Le code respecte les conventions de codage usuelles.
- La compilation selon vos instructions doit terminer sans erreur ni avertissement.
- Le `README.md` décrit les fonctionnalités effectivement implémentées et explique vos choix techniques (modélisation, patrons de conception, ... ).
- Un diagramme de classe rendra la modélisation plus claire !
- Le code est architecturé intelligemment en favorisant la réutilisation de code. Notamment, il faut utiliser des patrons de conception vus en cours ainsi que les constructions les plus appropriées fournies par Java.
- L'exécution doit être correcte : elle respecte le cahier des charges et ne quitte pas de façon non contrôlée.
- Les classes et méthodes sont documentées (javadoc).
- Des commentaires expliquent les portions de codes qui ne sont pas évidentes.
- Les tests fournis doivent être aussi exhaustifs que possible.

**La soutenance :**

- Des démos ont été préparées pour exposer les fonctionnalités du projet.
- Le programme doit pouvoir être compilé et exécuté sous les yeux du jury à partir de la version présente sur la branche principale de votre git (ou bien la dernière version avant la date limite).
- Vous ferez en sorte de pouvoir répondre à toute question concernant vos choix techniques. C'est-à-dire que non seulement vous saurez expliquer ce que vous avez fait, mais aussi ce que vous auriez-pu faire à la place et ce qui vous a mené à faire le choix que vous avez fait. N'hésitez pas à relire le cours et les TPs de CPOO5 en faisant le projet.