TRAVAUX DIRIGES DE SIMULATION ENSTA BRETAGNE - PARTIE 1 ANNEE 2019-2020

Enseignant : Olivier VERRON (simuenstabretagne@gmail.com)

1. PREAMBULE

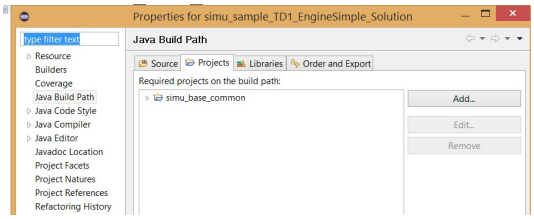
Cette partie du TD va vous conduire à apprendre à faire un moteur de simulation événementiel.

2. OBJET DE L'EXERCICE

- Créer un type d'objet « Moteur de simulation » et sa boucle de simulation
- Créer un type d'objet « Evénement de simulation »
- Créer un type d'objet « Entité de simulation »
- Créer une application Helloworld

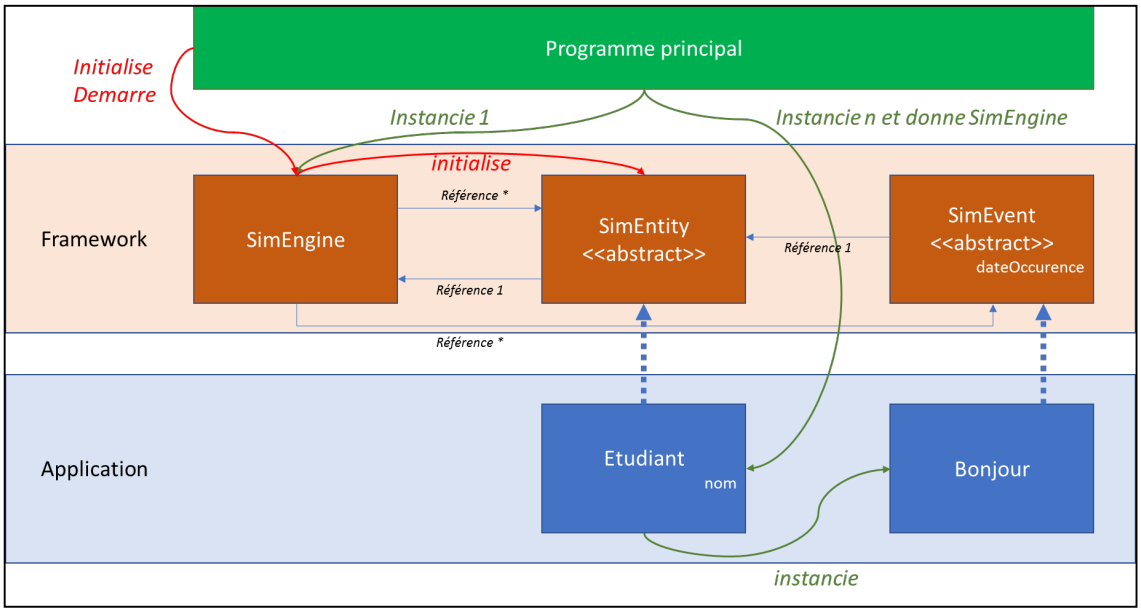
3. PREPARATION DE L'ESPACE DE TRAVAIL

- Créer un Projet Java
- Ajouter une dépendance vers simu_base_common



- Créer deux packages Java dans la source:
  - Un package dédié au framework du moteur de simulation : par exemple : `enstabretagne.travaux_diriges.EngineSimple.Framework`
  - Un package dédié à l'application HelloWorld : par exemple : `enstabretagne.travaux_diriges.HelloWorldSimple`

4. RAPPELS DU PRINCIPE DU FRAMEWORK



L'un des buts du framework est de factoriser au sein de classes abstraites ou concrètes des traitements récurrents entre classes du framework.

Le mécanisme d'héritage, de surcharge des méthodes permet alors de spécialiser les fonctions génériques du framework sur le besoin.
Ainsi, au moment de fabriquer le programme principal et l'application concrète, on fait hériter les éléments concrets à implémenter et spécifiques du besoin des classes abstraites du framework.

Généralement le programme principal a pour mission d'instancier le framework et de lui adjoindre la configuration d'exécution sur les classes concrètes liées au besoin.

Ci-dessus illustre le cas du présent TD.
Une des questions principales qui se pose quand on crée un framework est : qui fait quoi dans le cycle de vie des objets. La combinatoire rend l'exercice vite difficile. Il y a rarement des solutions clairement meilleures.

Ci-dessus vous illustre une proposition de séquencement :

- Qui référence qui
- Qui créé (flèche en vert) qui et dans quel ordre (flèches en rouge)

5. CREATION DU FRAMEWORK

Tout ce qui va suivre doit être réalisé dans
`enstabretagne.travaux_diriges.EngineSimple.Framework`

Vous êtes invités à lire tout le présent paragraphe avant de le traiter.

Créer 3 types d'objets vides dans :

- `SimEvent`
- `SimuEngine`
- `EntiteSimulee`

5.1 Créer une notion d'événement de simulation

L'événement de simulation sera un type d'objet fondamental. Par définition il a :

- Une date d'occurrence

- Porte sur une entité de simulation
- A une action, qu'on appellera par exemple par le fonction Process(). La créer vide ou abstraite.

5.2 Créer une notion d'entité de simulation

L'entité simulée par définition

- Possède des propriétés, qu'on peut par exemple transmettre via son constructeur. Mais au niveau du framework, on est générique, donc pas de propriétés.
- Elle va nécessiter d'être initialisée

5.3 Créer le moteur de simulation

Le moteur est celui qui régit le cycle de vie des entités. Dans notre exemple ultrasimple, on se limitera uniquement à la gestion de l'initialisation puis de l'exécution.

Il gère :

- le temps, par la fameuse boucle de simulation
- un échéancier de `SimEvent`,
- la liste des entités simulée qu'il a à initialiser,
- le début et la fin de la simulation

Une des difficultés de conception du framework est de faire en sorte que les 3 types d'objets fondamentaux se connaissent mutuellement lorsqu'ils seront instanciés.

Il vous est proposé :

- de faire connaitre le moteur à l'entité lors de la création de cette dernière, et que suite à cela, l'entité se fait connaitre au moteur
- de faire connaitre l'événement temporel d'abord à l'entité par le fait de lui attribuer l'événement, qui, elle, va ensuite faire connaitre l'événement au moteur, ce dernier le place alors dans l'échéancier

Penser que le moteur doit gérer dans l'ordre les événements. Il vous est donc proposé de rendre le `SimEvent` comparable avec lui-même afin que l'échéancier puisse le traiter à son tour.

6. CREATION DE L'APPLICATION HELLOWORLD SIMPLE

Le but de l'application est très simple :

Créer tout d'abord des étudiants qui disent bonjour « en l'air » sans personne pour répondre.

Le but du framework précédent est de ne pas avoir à recréer les liens entre le moteur, l'entité, les événements quand on fait l'application proprement dite. La mécanique est faite une fois pour toute.
On va donc **hériter** du framework pour créer l'application.

Tout ce qui va suivre doit être réalisé dans
`enstabretagne.travaux_diriges.HelloWorldSimple`

Créer 2 types d'objets

- L'événement `Bonjour` héritant de `SimEvent`
- L'entitée de simulation `Etudiant` héritant de `EntiteeSimulee`

6.1 Créer l'événement Bonjour !

Créer un constructeur qui prend en paramètre le nécessaire pour pouvoir dire bonjour par exemple le nom de la personne qui le dit, ou le message lui-même. Comme il hérite de `SimEvent`, il hérite aussi des arguments prévus d'être passés au constructeur de sa classe héritée.

Cette fois surcharger la fonction `Process()` et faite en sorte qu'il produise une trace à l'écran (par exemple en utilisant le logger) en disant Bonjour !

6.2 Créer l'entité simulée Etudiant

Créer un constructeur qui prend en paramètre le nécessaire pour créer l'étudiant.
L'initialisation est le seul moment où l'entité peut initier le premier bonjour… C'est donc là que l'étudiant doit prévoir de poster l'événement du Bonjour au moteur au moment absolu désiré.

6.3 Trouver un moyen de faire répéter le bonjour toutes les 5 min

Voir le corriger avec le projet `simu_sample_TD1_EngineSimple_Solution`

6.4 Pour aller plus loin

Vous pouvez commencer à réfléchir à comment mettre les entités en relation entre elles sans qu'elles aient à se connaitre à priori… et l'implémenter si vous le souhaitez.
Quel objet est le plus adapté à cette mise en relation ?
Comment formaliser cette mise en relation ?