Arguments variables - Classe Iterable

Resp. UE : Érik Martin-Dorel & Jean-Paul Bodeveix

Sujets POO: Christelle Chaudet & Jean-Paul Bodeveix

Préparation de l'environnement de développement

Dans les TPs nous utiliserons obligatoirement :

- l'IDE Eclipse avec le plugin SonarLint,
- l'historisation du développement avec Git.

Assurez-vous d'avoir les outils nécessaires (IDE Eclipse, terminal pour l'utilisation de GIT, un compte GitHub), sinon vous trouverez les documents d'installation sur Moodle.

Sujet

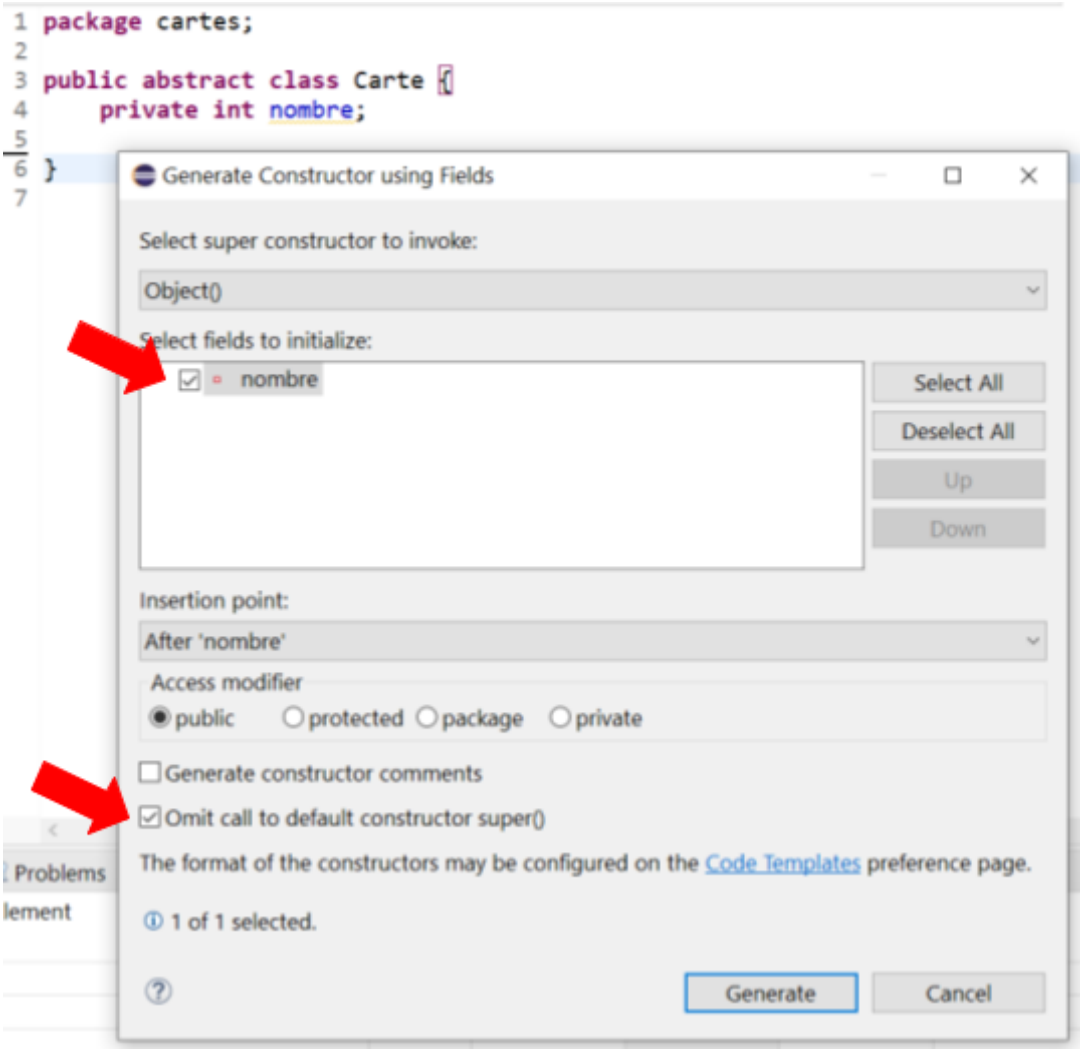
Cette série de TP a pour but de réaliser un jeu de milles bornes où les joueurs sont gérés par la machine selon diverses stratégies. Les règles du jeu sont décrites ici : règles du jeu. Les TPs introduisent progressivement les éléments constitutifs du jeu en exploitant les notions vues en cours.

Les Cartes

En utilisant les outils de génération automatique d'Eclipse, définir les classes du diagramme en page 4 avec leurs constructeurs et accesseurs en lecture.

Rappel :

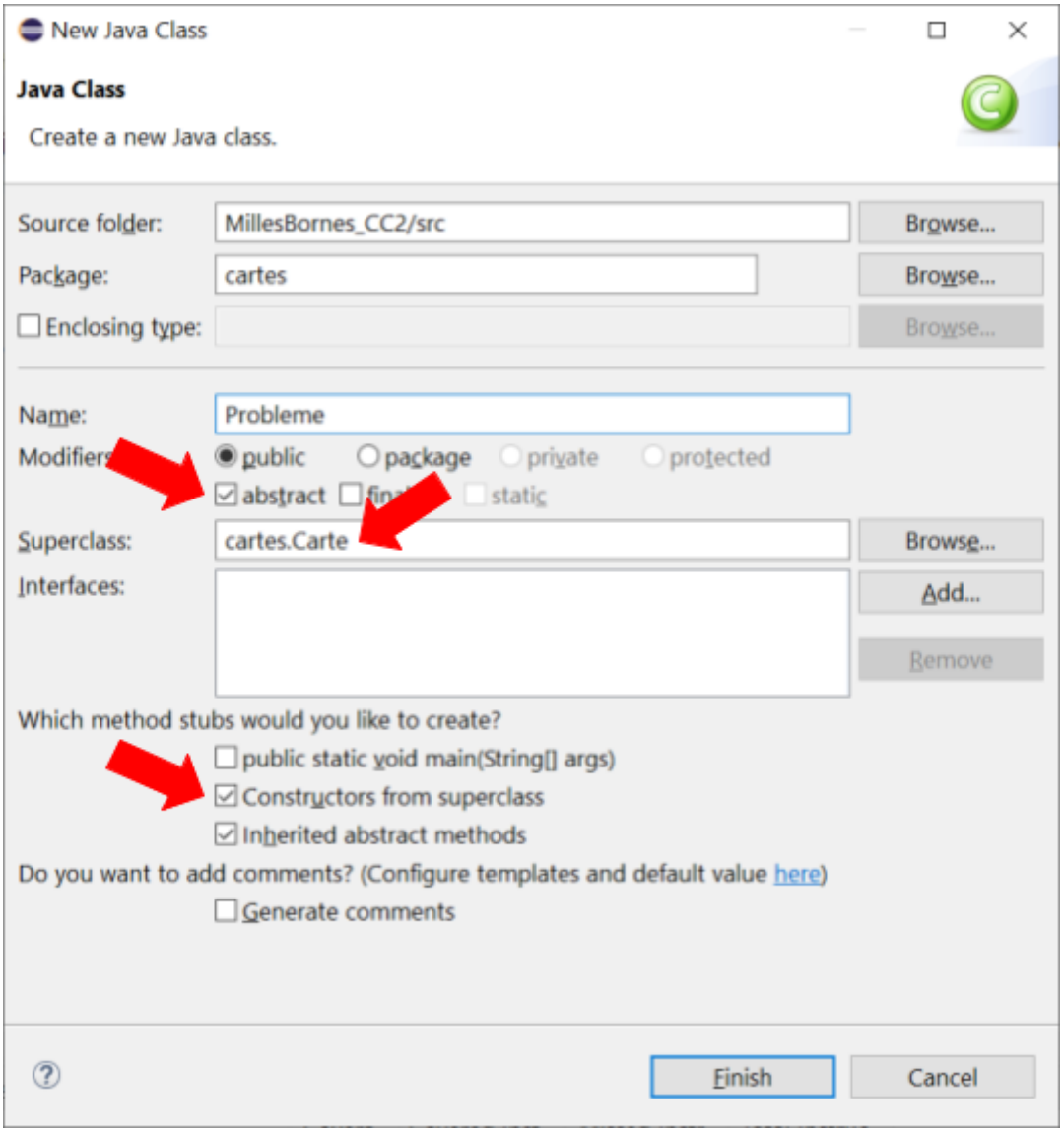
Utiliser la génération automatique des constructeurs et des des getters.



Pour les getters soit il y en a plusieurs et vous passez par Source > Generate Getters and setters... et cochez ce que vous souhaitez générer (les getters ou les setters ou les deux).

Soit il n'y en a peu et vous tapez directement dans votre code : get + Ctrl + espace

Lors de la création de la classe sélectionner les cases qui vous intéressent.



Sélectionner ou non abstract.

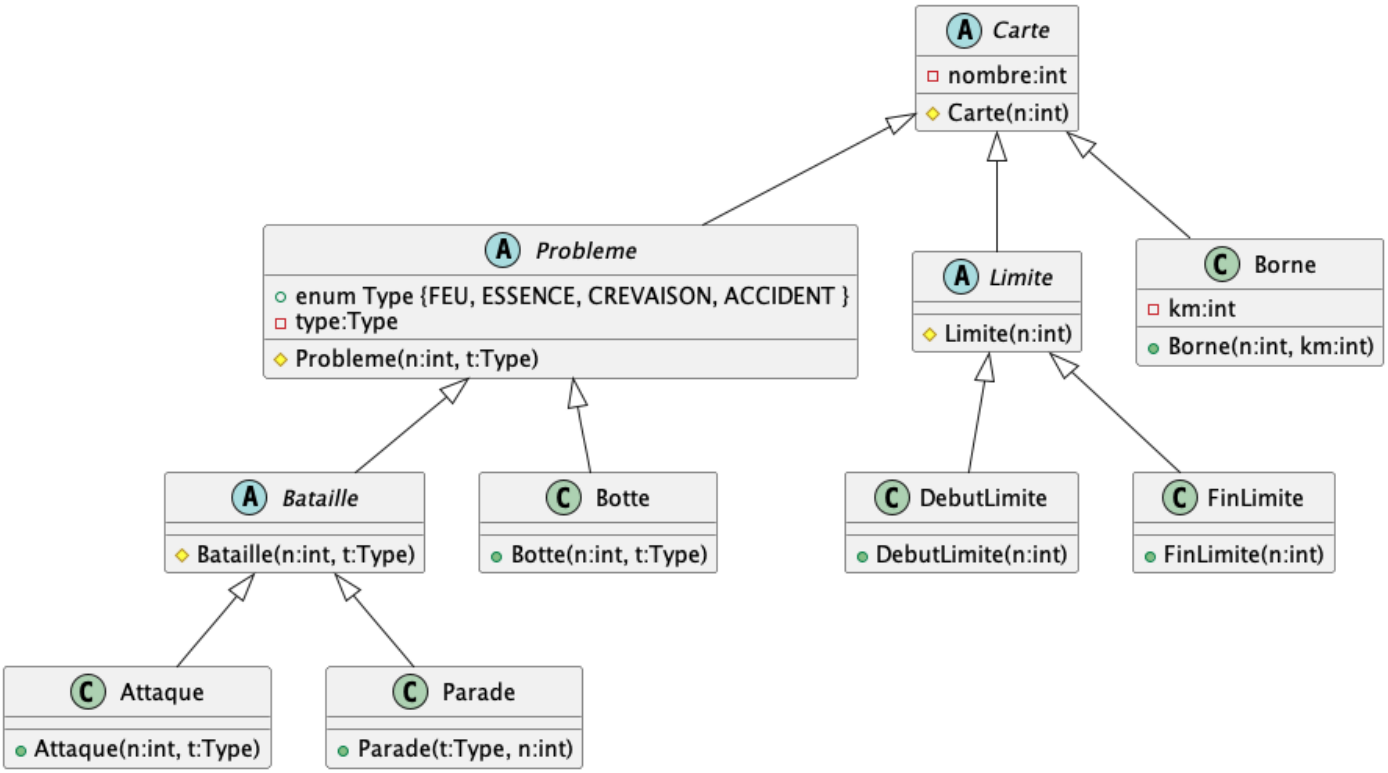
La superclasse est par défaut la classe Object, taper le début de la classe que vous voulez en classe mère (ex : Ca + Crt + espace), Eclipse trouvera le nom du paquetage et le nom de la classe.

Si la classe mère a un constructeur cocher “Constructors from superclass”.

Si la classe mère possède des méthodes abstraites cocher “Inherited abstract methods”

Travail à effectuer :

1. Implémenter les classes du diagramme ci-dessous.



2. Définir les méthodes **toString** renvoyant le nom de chaque carte en s'inspirant de la description du contenu d'un jeu donnée dans règles du jeu. On nommera FeuRouge et FeuVert les attaques et parades représentées par des feux. On considèrera que VéhiculePrioriataire est une botte associée au feu et non à une limite de vitesse.

Le Sabot

Cette partie est réalisable durant la séance de TP. Si vous ne l'avez pas terminée à la fin des deux heures vous devrez la terminer chez vous avant la séance suivante.

1. Ajouter un package jeu contenant la classe Sabot contenant des cartes stockées dans un tableau dont la capacité sera fournie par le constructeur (le jeu comporte 110 cartes). L'attribut nbCartes indiquera le nombre effectif de cartes du sabot.

    a. Définir la méthode **estVide** indiquant si la pioche est vide.
    b. Définir la méthode privée **ajouterCarte** ajoutant une seule carte. Une exception sera levée en cas de dépassement de capacité.
    c. Définir la méthode **ajouterFamilleCarte** ajoutant la carte passée en paramètre autant de fois que le nombre de cartes de cette famille l'indique.
    d. Définir la méthode **ajouterFamilleCarte** ajoutant les cartes passées en paramètre en respectant leurs nombres d'occurrences.
    e. Rendre la classe itérable: on gèrera les exceptions IllegalState et ConcurrentModification. La méthode remove devra être définie.
    f. Utiliser un itérateur pour définir la méthode piocher() renvoyant et supprimant la première carte du sabot.

2. Écrire dans un package **testsFonctionnels** un programme de test ajoutant les familles de cartes accident, réparation (parade d'accident) et asDuVolant (botte associée à accident) à un sabot et prélevant et affichant ses cartes jusqu'à ce qu'il soit vide.
    a. On utilisera piocher. L'affichage attendu est le suivant:

```
je pioche Accident
je pioche Accident
je pioche Accident
je pioche Réparation
je pioche Réparation
je pioche Réparation
je pioche As du volant
```

    b. On utilisera un itérateur et remove pour obtenir le même résultat
    c. On ajoute à la boucle b) un appel à piocher et l'"ajout d'asDuVolant. Une exception doit être levée.