IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

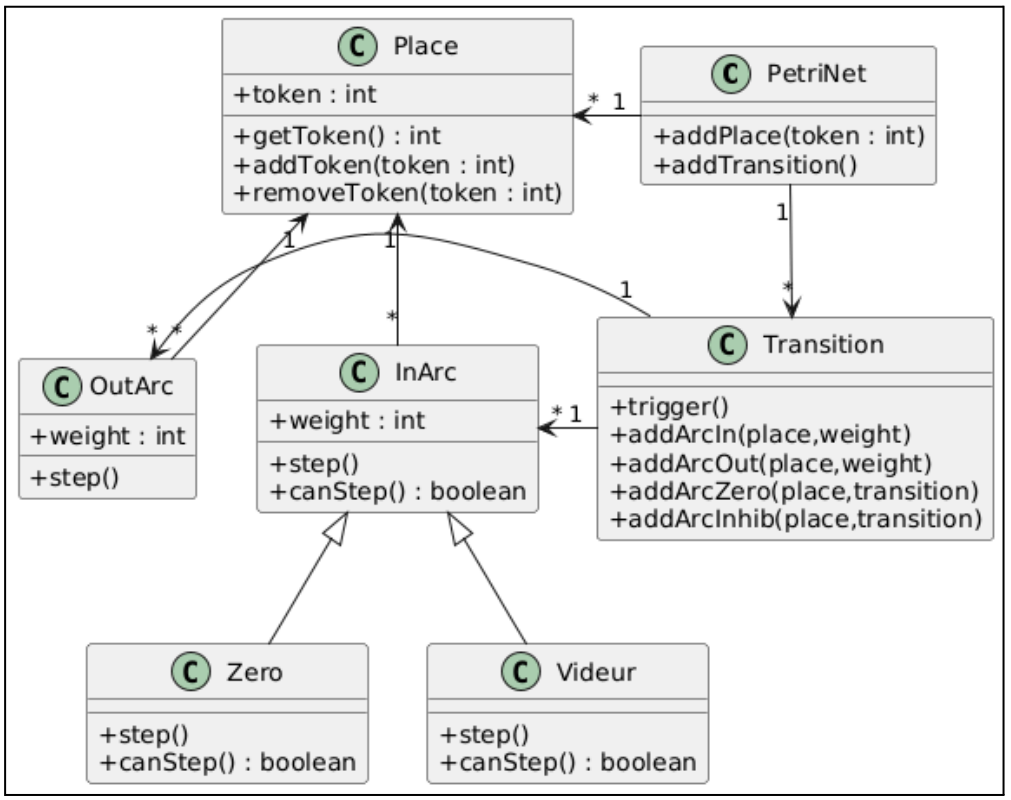
MAPD:

Conception finale

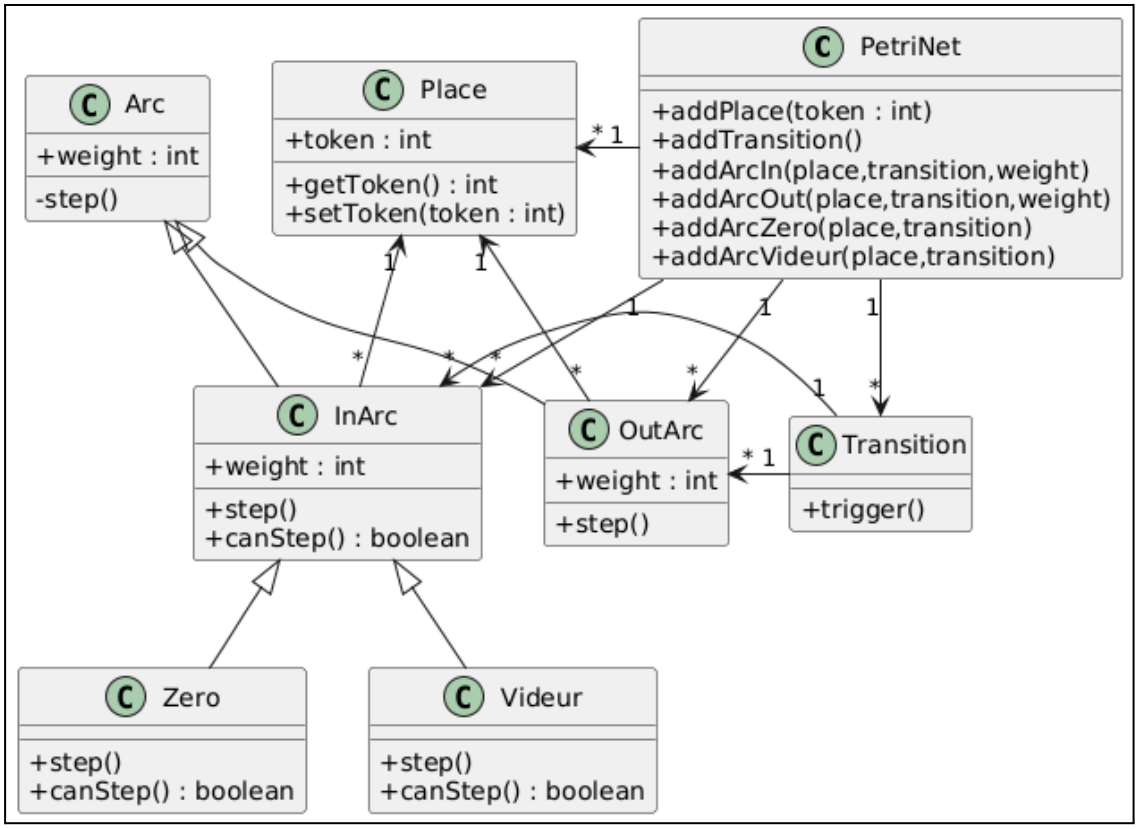
XU Johan, Karikalan Sanchjeev

Diagramme de classe détaillé:

retenu :



variante :



PetriNet

Nous avons décidé de créer une classe “PetriNet” qui va nous permettre d’ajouter:
-des places, chacune contenant un nombre de jetons défini,
-des transitions

Place

Les places possèdent un attribut “token” qui indique le nombre de jetons que la place possède, une méthode getToken() qui renvoie le nombre de jetons de la place, une méthode **addToken(**token:int**)** qui va nous permettre d’ajouter des jetons à la place et une méthode **removeToken(**token:int**)** qui en enlever à la place.

InArc

Les arcs entrant dans une transition possèdent un attribut “weight” qui indique son poids, une méthode canStep() qui va vérifier que le tirage est bien possible pour cette arc en comparant son poids et le nombre de jetons de la place auquel il est lié et une méthode step() qui va effectuer le tirage en retirant les jetons de la place auquel il est lié grâce à la méthode removeToken() de place.

ArcZero et ArcVideur

Ce sont des cas particuliers d’arcs entrant qui vont hériter de la classe mère “InArc”, leurs méthode “canStep()” va être redéfini car leurs conditions d’activation sont différents

OutArc

Les “OutArc” sont les arc sortants des transitions. Cette classe possède un poids et une méthode step() qui va être appelée par la classe “Transition”, si le tirage est possible la méthode va donc ajouter l’équivalent de son poids en jeton à l’arc auquel il est relié en utilisant la méthode addToken() de la place.

Transition

Cette classe possède une méthode trigger() qui va vérifier que les conditions pour tirer sont réunies. Elle va demander à tous les “InArc” si le step est faisable (canStep()) et réaliser le tirage si tous les “inArc” peuvent réaliser un step, en appelant la méthode step() des “InArc” et “OutArc”.
Cette classe a des méthodes pour ajouter les différents types d’arcs avec leurs poids et les places auxquelles ils sont reliés.

Comparaison des diagrammes de classes

Nous avons pas retenu la solution variante en raison de trois différences:

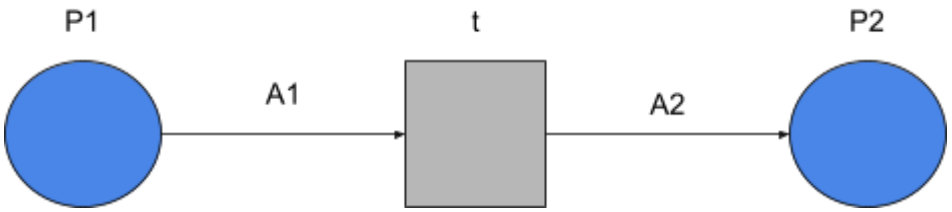
-Dans le diagramme de classe variant, il y a une classe mère Arc dont hérite la classe OutArc et InArc, avec un attribut “weight” et une méthode “step()”. Cependant nous avons décidé de ne pas conserver cette classe, puisqu’elle nous semble peu utile, mise à part pour l’attribut “weight”. En effet, les méthodes step() seraient à redéfinir dans InArc et OutArc car elles n’ont pas la même fonctionnalité. Une possibilité aurait été de mettre en place une classe abstraite.

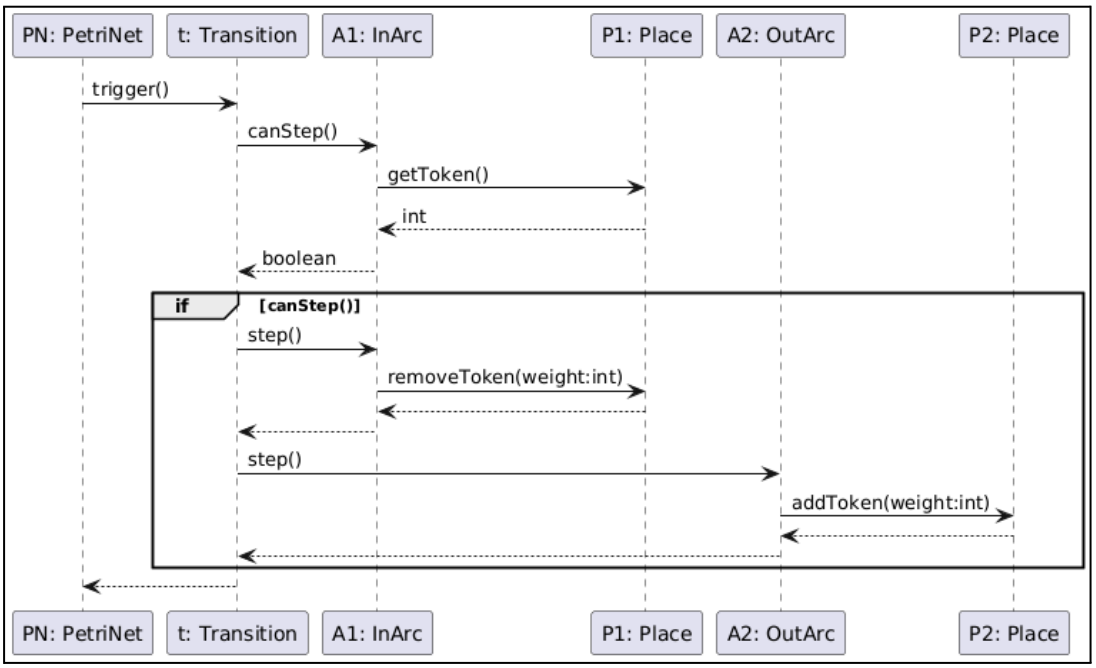
-La classe Place possède des méthodes différentes pour modifier le nombre de jetons. (setToken / addToken et removeToken). Nous avons décidé de conserver le choix addToken et removeToken, car pour utiliser setToken, il faut que l’arc fasse un appel à la méthode getToken. Ainsi, on évite un autre aller-retour inutile dans le diagramme de séquence.

-Dans le diagramme de classe retenu nous avons mis les méthodes qui permettent d’ajouter les différents types d’arcs dans la classe transition, contrairement à la variante où les méthodes sont dans la classe PetriNet, car le PetriNet n’a pas besoin de connaître les arcs, mais seulement les places et les transitions. Il suffit juste que les transitions connaissent les arcs.

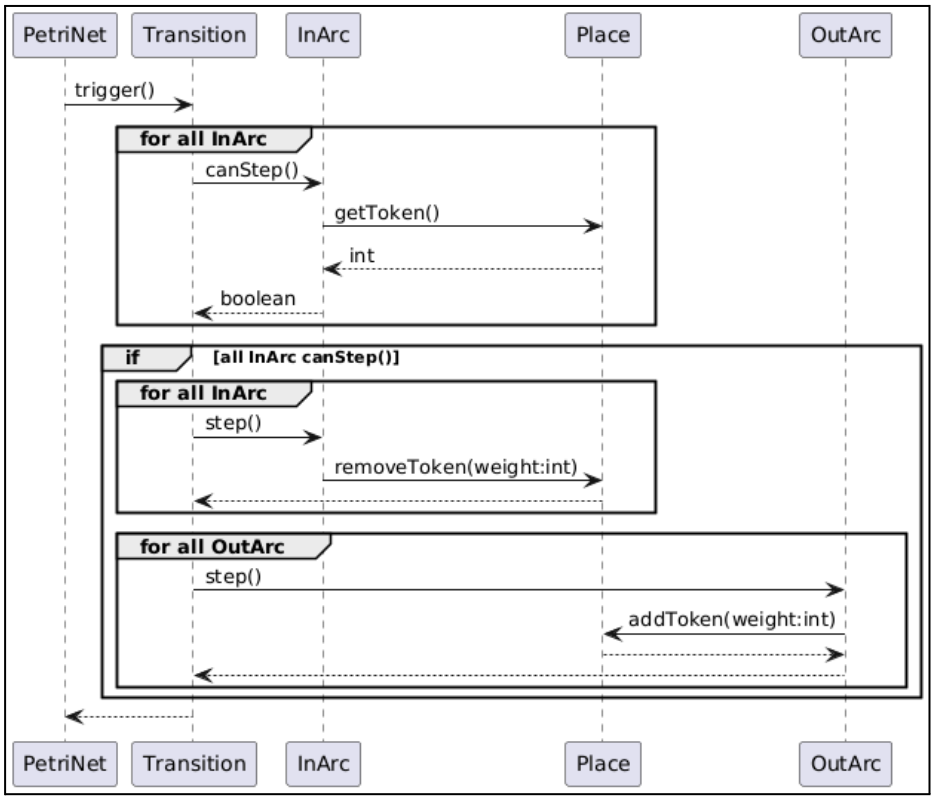
Diagramme de séquence:

cas simple **:**





cas général :



Le diagramme de séquence modélise le processus de création et de manipulation d'un réseau de Petri.

petrinet -> transition : le réseau vérifie si une transition peut être effectué.

Transition -> InArc: canStep(), InArc -> Place: getToken(), Place --> InArc: int, InArc --> Transition: boolean : Cette série d'interactions vérifie si les places connectées à la transition ont suffisamment de jetons pour permettre le tirage.

Le groupement “if can step” modélise le cas où la transition peut être déclenchée.

Transition -> InArc: step(), InArc -> Place: removeToken(weight) : Si la transition est effectué, les arcs d'entrée retirent des jetons des places connectées.
Transition -> OutArc: step(), OutArc -> Place: addToken(weight) : Ensuite, les arcs de sortie distribuent des jetons dans les places de sortie.