ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

TP 4
Héritage & planche à clous

Objectifs

Héritage. Généralisation et spécialisation. Redéfinition de fonctions membres. Méthodes virtuelles et polymorphisme.

Durée : 2 séances

Vous allez dans ce TP programmer un simulateur pour le parcours d'un palet au travers d'une planche à clous, telle que celle qui est représentée sur la figure 1. Le but avoué de tout ceci est de calculer, de manière plutôt indirecte, les coefficients du binôme...

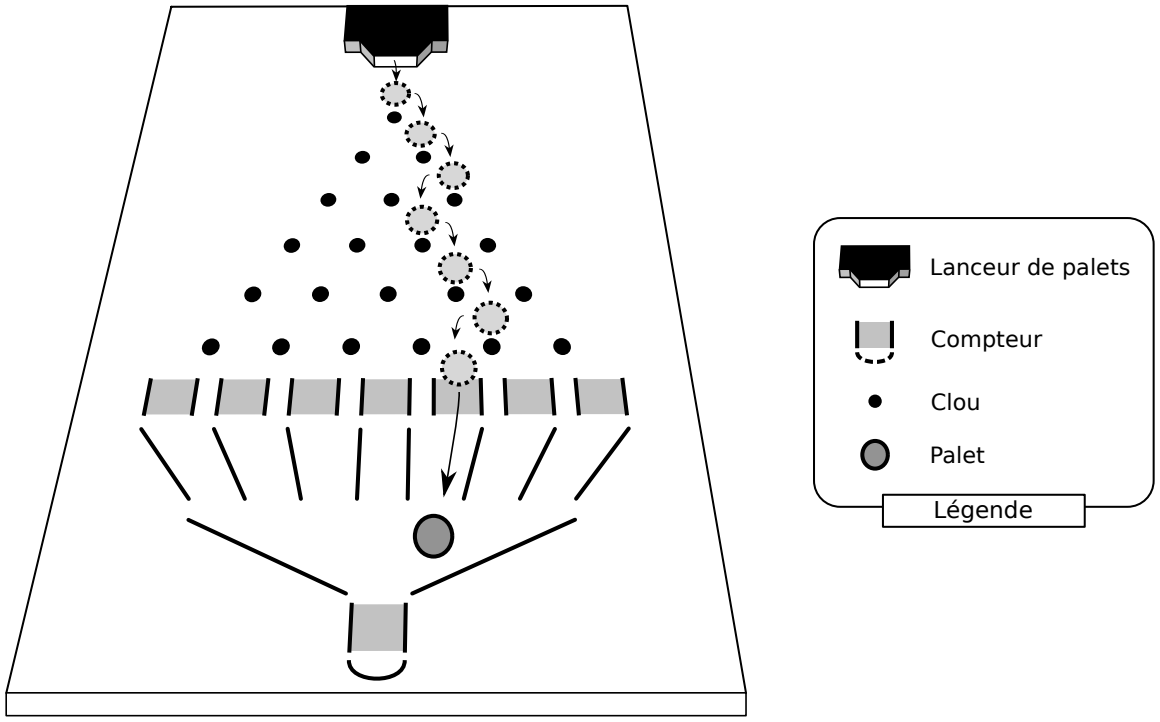

FIGURE 1 – Vue d'ensemble de la planche à clous.

1 Présentation

Vous devez programmer toutes les classes qui modélisent les composants de cette planche à clous, en suivant le diagramme de classes qui vous est proposé dans la figure 2.

Un obstacle est, d'une manière générale, un élément qui peut recevoir le palet. Ce peut être un clou ou un compteur. Un clou qui reçoit un palet le transmet, de manière équiprobable, soit à l'obstacle qui est sous lui à sa gauche, soit à celui qui est à sa droite. Un compteur qui reçoit un palet incrémente une donnée membre (qu'il est possible de consulter ou remettre à zéro) et transmet le palet à l'obstacle suivant, s'il y en a un (cf. figure 1). Enfin, un lanceur offre comme service le fait d'envoyer un palet à un obstacle. En fait, le palet n'existe qu'au travers de la méthode `reçoitPalet()`.

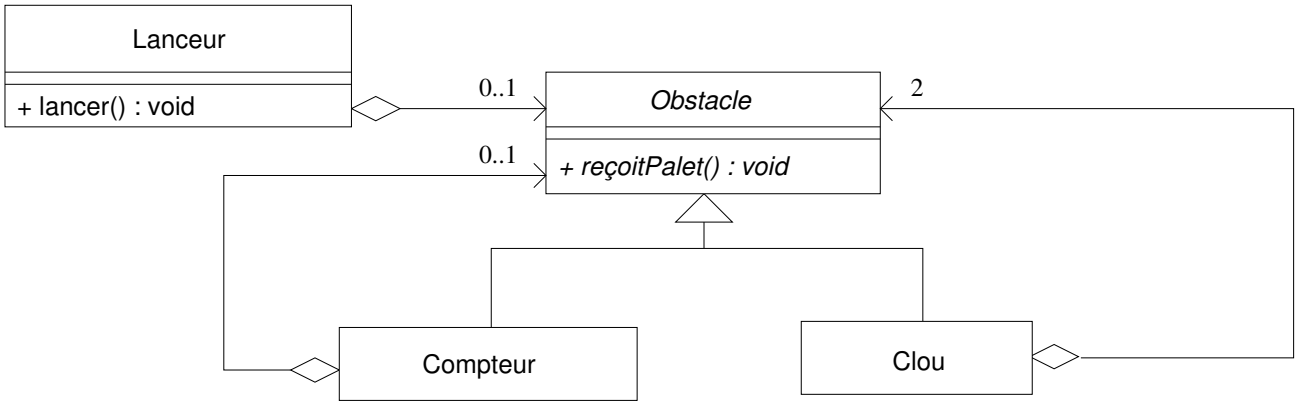

FIGURE 2 – Diagramme de classes de la planche à clous.

NOTE IMPORTANTE Les classes de ce TP ne sont pas destinées à être copiées (affectation, passage d'argument par valeur, etc.). Il ne vous est donc pas demandé de programmer les méthodes correspondantes. En fait, il faut même empêcher tout copie en désactivant les comportements par défaut (voir la syntaxe « `= delete` », section 2.1.9 du polycopié de cours).

2 Construction de la simulation

Pour construire la planche à clou (dans le constructeur de la classe Planche, cf. section 4.1), vous commencerez par créer un tableau de pointeurs vers des compteurs [1], utile pour « relever les compteurs » en fin de simulation. Tous ces compteurs auront comme obstacle suivant un unique compteur sans successeur. Ce dernier servira à vérifier que le nombre de palets lancés correspond au nombre total de palets arrivés dans les compteurs du dernier rang.

Ensuite, les différents niveaux de clous seront construits de bas en haut en $N-1$ itérations, où N est le nombre de compteurs du dernier rang. À chaque itération k, si O_{k-1} désigne le nombre d'obstacles de l'itération précédente, alors on crée $O_{k-1}-1$ clous. Le clou de position i, $i \in \{0,\ldots,O_{k-1}-2\}$, aura pour successeurs gauche et droit les obstacles de positions i et $i+1$ du rang $k-1$.

3 Principe de la destruction

La construction de la planche étant dynamique (le nombre de compteurs peut n'être connu que lors de l'éxécution), l'allocation des différents objets sera faite sur le tas. Il s'agit donc de gérer correctement la desallocation de tous les objets depuis le destructeur de la classe Planche (cf. section 4.3).

Ici, les obstacles concrets (c.-à-d. les clous et compteurs) et le lanceur se chargeront, *sauf exception*, de la désallocation des obstacles qui les suivent.

L'exception évoquée plus haut provient de la situation illustrée dans la figure 3 où un clou est le suivant de deux autres clous. En respectant à la lettre la règle énoncée, de nombreux obstacles seraient désalloués deux fois, ce qui n'est pas envisageable ! En fait, tous les obstacles « intérieurs » de la planche ainsi que le dernier compteur sont dans ce cas. Afin d'apporter une solution à ce problème, vous définirez la notion d'obstacle *orphelin*. Un obstacle aura le statut d'orphelin (un booléen) si aucun autre objet n'est chargé de le désallouer. Ainsi, un Lanceur, un Compteur ou un Clou, peuvent durant leur construction interroger leurs successeurs. S'ils sont orphelins, l'objet s'attribue la responsabilité de la désallocation du successeur

1. Les compteurs seront alloués un par un sur le tas, comme tous les obstacles de la simulation. C'est une conséquence de la méthode de libération expliquée dans la section 3.

en question, et lui signifie qu'il n'est plus orphelin. De cette façon, deux objets ne pourront s'attribuer la responsabilité de la désallocation d'un même obstacle.

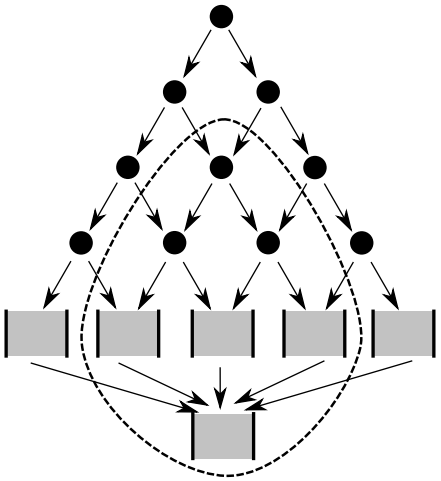

FIGURE 3 – Ensemble (en pointillés) des obstacles ayant plus d'un prédécesseur.

Question 1

Les associations (agrégations) Clou-Obstacle et Compteur-Obstacle définissent en pratique un *graphe orienté* d'obstacles (que l'on peut visualiser sur la figure 3). Dans ce contexte, la relation « père-fils » établie pour la désallocation correspond à un sous-ensemble d'arêtes, et donc un sous-graphe. Quel nom porte ce type de sous-graphe vis-à-vis du graphe de départ ?

4 Travail à réaliser

4.1 Classe Planche, construction

Programmez une classe `Planche` qui regroupera toutes les données propres à une simulation (lanceur, nombre de compteurs, tableau de pointeurs vers les compteurs, compteur terminal). Le constructeur de cette classe sera paramétré par le nombre N de compteurs du dernier rang. Le cas limite à un seul compteur correspondra à un lanceur directement relié à cet unique compteur.

4.2 Simulation

Programmez une *méthode* (classe `Planche`) de simulation paramétrée par le nombre de palets à lancer. En fin de simulation, la fonction affichera les valeurs suivantes :

$$round\left(\frac{c_0}{c_0}\right),\ round\left(\frac{c_1}{c_0}\right),\ round\left(\frac{c_2}{c_0}\right),\ldots,round\left(\frac{c_{N-1}}{c_0}\right)$$

où c_i, pour $i \in \{0, N-1\}$, est la valeur du compteur en position i dans le tableau des pointeurs de compteurs créé lors de la construction de la planche (cf. 4.1). Vérifiez que les valeurs obtenues coïncident avec les coefficients binomiaux (Table 1 pour $N \leq 5$).

N	$\binom{N}{k}_{k\in\{0,\dots,n\}}$					
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

TABLE 1 – Les 6 premières lignes du triangle de Pascal.

4.3 Classe Planche, destruction

Lors de la destruction d'une planche, la destruction complète de tous les obstacles sera provoquée récursivement par la destruction du lanceur.

Question 2

Quelle propriété devra posséder le destructeur de la classe des obstacles pour assurer un fonctionnement correct de la destruction récursive évoquée ci-dessus ?

4.4 Vérification

Vérifiez que votre simulation ne provoque pas de fuite mémoire à l'aide de l'outil valgrind.

5 Améliorations (bonus, pour les plus avancés)

— Ajouter un compteur (statique) d'instances pour vérifier que tous les objets sont bien détruits.
— Afficher sur la sortie standard la position courante du palet (avec une marge faite d'espaces) à chaque fois que ce dernier touche un obstacle.
— Afficher le plateau complet.