UE Informatique
Étape 3 - La pile de planchettes et son équilibre

Olivier Beaudoux (olivier.beaudoux@eseo.fr)

Semestre 4

Table des matières

1 La pile de planchettes

Les planchettes sont prises dans les pioches des joueurs pour constituer, jusqu'à son déséquilibre, la pile de planchettes. Comme abordé lors de l'étape 2 précédente, le type abstrait *Planchette* représente une planchette *immuable* de longueur et de marge données. Les données nécessaires au calcul de l'équilibre de la pile sont reléguées au type abstrait *Empilement* ; elles sont modifiées au fur et à mesure des empilements successifs de planchettes.

La figure 1 fournit un modèle[1] des 3 types abstraits *Pile*, *Empilement* et *Planchette* à considérer dans cette étape.

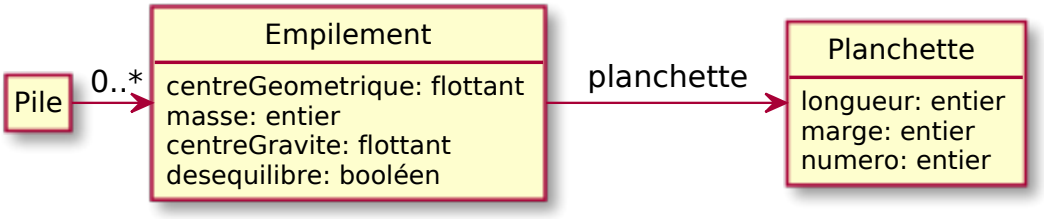

FIGURE 1 – Modèle des empilements

Une pile est constituée de 0 à plusieurs (symbole $*$) empilements. Un empilement décrit les données relatives à l'empilement d'une planchette. Le type abstrait *Planchette* a été implémenté lors de l'étape 2, le type *Empilement* est abordé dans la section 1.1 et le type *Pile* dans la section 1.2. Le calcul des masses et des centres de gravité est abordé dans la section 2 en complétant l'implémentation du type *Pile*.

1.1 Empilements

Introduction

La figure 2 montre une succession de six *empilements* successifs de planchettes numéro 181 avec un décalage de 2 cm.

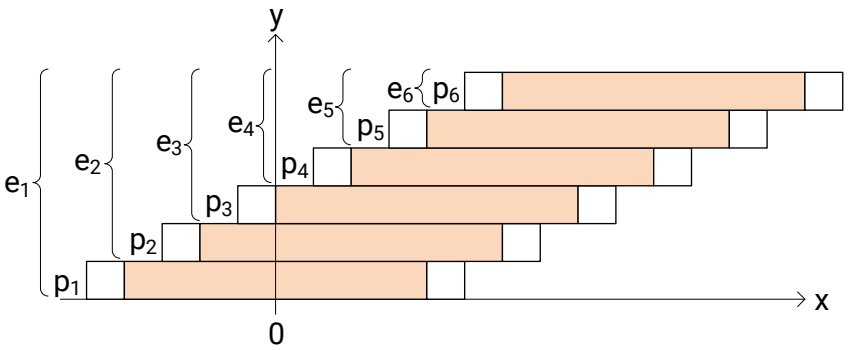

FIGURE 2 – Exemple d'empilements

1. Il s'agit d'un diagramme de classes écrit en PlantUML.

Le premier empilement réalisé est situé en $y = 0$ et définit le centre $x = 0$ de la pile : les coordonnées de tous les empilements réalisés ultérieurement sont donc relatifs à ce premier empilement.

Un empilement e_i contient l'ensemble des planchettes p_j pour j variant de i à n où n est le nombre total de planchettes, soit $e_i = \{p_j | j \in [i..n[\}$. Chaque nouvel empilement effectué à un instant t modifie les empilements précédents :

- $t = 0$: la pile est vide
  - $pile = \emptyset$
- $t = 1$: empilement e_1 de la première planchette p_1
  - $e_1 = \{p_1\}$
  - $pile = e_1$
- $t = 2$: empilement de la première planchette p_2
  - $e_2 = \{p_2\}$
  - $e_1 = \{p_1, p_2\}$
  - $pile = e_1$ invariablement
- $t = 3$: empilement de la première planchette p_3
  - $e_3 = \{p_3\}$
  - $e_2 = \{p_2, p_3\}$
  - $e_1 = \{p_1, p_2, p_3\}$
- et ainsi de suite.

L'empilement e_i représente donc toutes les planchettes au dessus et à partir de la planchette p_i. Le principe de l'équilibre de la pile peut maintenant s'énoncer comme suit :

> « L'empilement e_i met la pile en déséquilibre si est seulement si son centre de gravité est situé *strictement* à gauche ou à droite du bord de la planchette p_{i-1}. »

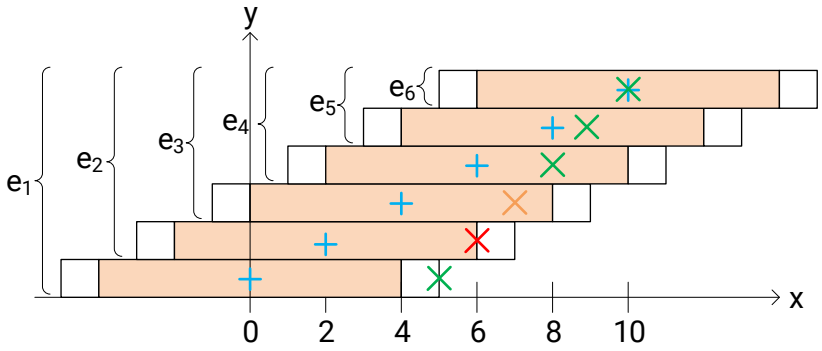

FIGURE 3 – Centres et équilibre d'un empilement

La figure 3 utilise le symbole $+$ pour représenter le centre géométrique de chaque empilement e_i (correspondant au centre de la planchette p_i) et le symbole $\times$ représente son centre de gravité (aussi appelé centre des masses). Les deux centres coïncident pour une planchette seule, donc pour l'empilement supérieur e_6. L'empilement e_3 est en équilibre instable car son centre de gravité coïncide exactement avec le bord droit de la planchette p_2. Par contre, l'empilement e_2 est en déséquilibre car son centre de gravité

est strictement à droite du bord droit de la planchette p_1. L'empilement e_1 est toujours en équilibre car il ne repose pas sur une planchette mais sur une table (symbolisé par la droite $y = 0$).

Un empilement en déséquilibre met toute la pile en déséquilibre. Il est possible d'arriver à une situation où plusieurs empilements sont simultanément en déséquilibre. Un tel cas sera abordé dans la section 2.

Spécification

La spécification du type abstrait *Empilement* est la suivante :

***cree(planchette, centre)* :** retourne une structure de données qui représente l'empilement d'une *planchette* occupant une position précisée par l'abscisse (en cm) de son *centre* géométrique. La structure de données définit également la « masse » de l'empilement, initialement égale à la longueur de la planchette, l'abscisse de son centre de gravités, initialement égal à l'abscisse du centre géométrique, ainsi qu'un indicateur de déséquilibre initialisé à faux.

***planchette(empilement)* :** retourne la *planchette* de l'*empilement*.

***centreGeometrique(empilement)* :** retourne l'abscisse (en cm) du *centre géométrique* de l'*empilement*.

***masse(empilement, valeur=None)* :** modifie la « *masse* » (sans unité) de l'*empilement* en lui donnant la nouvelle *valeur*, dans le cas où cette valeur est définie ; retourne la « *masse* » de l'*empilement*. La masse correspond à la somme des longueurs des planchettes constituant l'empilement. Elle est mise à jour après chaque empilement lors du calcul du centre de gravité.

***centreGravite(empilement, valeur=None)* :** modifie l'abscisse du *centre* de *gravité* de l'*empilement* en lui donnant la nouvelle *valeur* (en cm), dans le cas où cette valeur est définie ; retourne l'abscisse (en cm) du *centre* de *gravité* de l'*empilement*. Le centre de gravité est mise à jour après chaque empilement.

***desequilibre(empilement, valeur=None)* :** modifie l'indicateur de *déséquilibre* de l'*empilement* en lui donnant la nouvelle *valeur*, dans le cas où cette valeur est définie ; retourne l'indicateur de *déséquilibre* de l'*empilement*. La propriété est mise à jour après chaque empilement.

***versChaine(empilement)* :** retourne une chaîne de caractères représentant textuellement l'empilement. Le format est le suivant : « n°n_i `m=`m_i `c=`c_i `g=`g_i » suivi du caractère '!' dans le cas où l'empilement est en déséquilibre. Par exemple, l'empilement e_i d'une planchette 262 avec $m_i = 10$, $c_i = 2$, $g_i = 3$ et en déséquilibre, est représenté par la chaîne « `n°262 m=10 c=2 g=3 !` ».

Travail à faire

Étape 3.1.1

- Télécharger depuis le campus l'archive « Etape-3.zip » et en extraire les fichiers dans le répertoire « S4-Informatique ».
- Copier les fichiers « Planchette.py », « Fenetre.py » et « VuePlanchette.py » de l'étape 2 précédente dans le répertoire « S4-Informatique/Etape-3 ».
- Compléter le fichier « Empilement.py ».
- Tester ce code en exécutant le module « TestEmpilement.py ».

Bien comprendre :

- **le principe de l'empilement ;**
- **le rôle de chaque propriétés modifiables du type Empilement.**

1.2 Pile

Cette section vise à implémenter les fonctions de base du type abstrait *Pile*, hors du calcul de l'équilibre de la pile et de ses empilements successifs.

Spécification

La spécification du type abstrait *Pile* est la suivante :

***cree()* :** retourne une liste initialement vide. Cette liste contiendra les empilements, le premier empilement effectué (bas de la pile) étant le premier de liste (indice 0) et le dernier empilement (sommet de la pile) étant le dernier de la liste (indice « taille-1 »).

***estVide(pile)* :** retourne vrai si la pile est vide, faux sinon.

***sommet(pile)* :** renvoie l'empilement au *sommet* de la pile. Dans le cas d'une pile vide, la valeur « None » est renvoyée.

***empile(pile, planchette, decalage)* :** crée un nouvel empilement et l'ajoute au sommet de la *pile*. La *planchette* ainsi que le *decalage* de son centre par rapport au centre de la planchette juste en dessous caractérisent l'empilement créé. La fonction ne vérifie pas la règle d'empilement concernant le décalage minimum ; elle ne vérifie également pas le premier empilement doit avoir un décalage nécessairement nul.

***versChaine(pile)* :** retourne une chaîne de caractères représentant textuellement le contenu de la pile. Par exemple, la représentation textuelle d'une pile contenant à sa base deux planchettes 262 et une planchette 545 à son sommet, avec les décalages successifs de 2 et 3 cm, est (voir le module *TestPile*) :

```
----------------------
n°545 m=14 c=5 g=5
n°262 m=10 c=2 g=2
n°262 m=10 c=0 g=0
^^^^^^^^^^^^^^^^^^^^^^
```

Travail à faire

Étape 3.1.2

- Compléter le fichier « Pile.py ».
- Tester ce code en exécutant le module « TestPile.py ».

Bien comprendre :

- **l'ordre du stockage des empilements dans le liste Python, le sommet de la pile étant le dernier élément de la liste ;**

- **le calcul du centre géométrique de l'empilement en fonction du décalage souhaité et du centre géométrique de l'empilement précédemment au sommet.**

2 L'équilibre des planchettes

2.1 Calcul des centres de gravité

Introduction

Chaque empilement $e_{i=1..n}$ est caractérisé par le numéro de la planchette p_i formant la base de l'empilement, sa masse totale m_i, l'abscisse c_i de son centre géométrique et l'abscisse g_i de son centre géométrique.

Chaque planchette p_i a une masse correspondant à sa longueur l_i [2]. La masse m_i de l'empilement e_i correspond donc à la somme des longueurs des planchettes $p_{j=i..n}$ formant l'empilement, soit :

$$m_i = l_i + l_{i+1} + ... + l_n$$

Chaque nouvel empilement modifie la masse des tous les empilements précédents. Il est donc plus simple de considérer la relation de récurrence entre les masses de deux empilements successifs, soit :

$$m_i = l_i + m_{i+1} \tag{1}$$

avec :

$$m_n = l_n \tag{2}$$

La position g_i du centre de gravité de la planchette i correspond à la moyenne des positions des centres géométriques $c_{j=i..n}$ des planchettes, chacune des positions étant pondérée par la masse l_i de la planchette. On peut ainsi écrire :

$$g_i = \frac{l_i c_i + l_{i+1} c_{i+1} + ... + l_n c_n}{m_i}$$

On peut alors montrer que la position g_i se détermine à partir de la position g_{i+1} comme suit :

$$g_i = \frac{l_i c_i + m_{i+1} g_{i+1}}{m_i} \tag{3}$$

avec :

$$g_n = c_n \tag{4}$$

Le formules (2) et (4) traduisent le fait que la masse (respectivement le centre de gravité) d'un nouvel empilement est initialisée avec la longueur (respectivement le centre géométrique) de la planchette associée (voir la spécification du type abstrait *Empilement* section 1.1).

2. Le coefficient de proportionnalité n'intervient pas dans le calcul du centre de gravité. Il est donc considéré comme égal à 1.

Spécification

La spécification complète du type abstrait *Pile* comprend la spécification donnée dans la section 1.2 complétée par les définitions suivantes :

empileEtCalcule(pile, planchette, decalage) : réalise l'empilement de la *planchette* au sommet de la *pile*, puis recalcule les centres de gravité et les éventuels déséquilibres de tous les empilements de la *pile*.

calculeCentresGravite(pile) : recalcule les centres de gravité de tous les empilements de la *pile*.

Travail à faire

Étape 3.2.1

- Implémenter la fonction *calculeCentresGravite* en se basant sur les formules (1) et (3).
- Tester cette implémentation en exécutant les modules « TestEquilibre1.py » et « TestEquilibre2.py ».

Indications :

- **Plusieurs variables locales doivent être clairement définies dans l'implémentation du calcul des centres de gravité. Le code doit être lisible et facilement analysable au regard formules mathématiques (1) à (4).**
- **Le premier test correspond aux empilements de la figure 3 page 3 ; l'affichage de la pile dans la console doit donner :**

```
----------------------
n°181 m=10 c=10 g=10
n°181 m=20 c=8 g=9.0
n°181 m=30 c=6 g=8.0
n°181 m=40 c=4 g=7.0
n°181 m=50 c=2 g=6.0
n°181 m=60 c=0 g=5.0
^^^^^^^^^^^^^^^^^^^^^^
```

- **Le second test doit donner l'affichage suivant :**

```
----------------------
n°464 m=14 c=0 g=0
n°262 m=24 c=3 g=1.25
n°464 m=38 c=6 g=3.0
n°262 m=48 c=9 g=4.25
n°464 m=62 c=12 g=6.0
n°262 m=72 c=14 g=7.111111111111111
n°464 m=86 c=11 g=7.744186046511628
n°262 m=96 c=13 g=8.291666666666666
n°464 m=110 c=10 g=8.50909090909091
```

```
n°262 m=120 c=8 g=8.466666666666667
n°464 m=134 c=5 g=8.104477611940299
n°262 m=144 c=3 g=7.75
n°464 m=158 c=0 g=7.063291139240507
^^^^^^^^^^^^^^^^^^^^^^^
```

2.2 Détermination des déséquilibres

Introduction

La figure 3 page 3 illustre le principe du déséquilibre de la pile de planchette induit par le déséquilibre de l'empilement e_2 (l'empilement e_3 est considéré en équilibre, même si cet équilibre est instable).

Un empilement e_{i+1} est en déséquilibre dès lors que le centre de gravité de cet empilement dépasse le bord droit ou gauche de la planchette à la base de l'empilement e_i situé juste en dessous, comme le montre la figure 1.

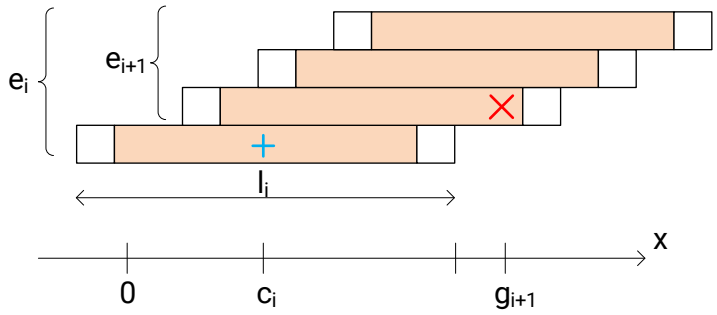

FIGURE 1 – Principe du déséquilibre

Le bord droit de la planchette à la base de e_i est situé à l'abscisse $b_i = c_i + (l_i/2)$ et l'abscisse g_{i+1}vérifie $g_{i+1} > b_i$: l'empilement $e_i$est en déséquilibre. On en déduit qu'il y a déséquilibre à droite dès lors que :

$$g_{i+1} > c_i + \frac{l_i}{2}$$

ce qui se généralise pour les bords droit et gauche en :

$$|g_{i+1} - c_i| \quad > \quad \frac{l_i}{2} \tag{5}$$

Spécification

La spécification complète du type abstrait *Pile* comprend une dernière fonction définie comme suit :

calculeEquilibre(pile) : recalcule les éventuels déséquilibres de tous les empilements de la *pile*.

Travail à faire

Étape 3.2.2

- Implémenter la fonction *calculeEquilibre* en se basant sur la formule (5).
- Tester cette implémentation en exécutant à nouveau les modules « TestEquilibres1.py » et « TestEquilibres2.py ». Les empilements en déséquilibre doivent pouvoir maintenant être identifiés.

Indications :

- **Tous comme pour le calcul des centre de gravité, différentes variables locales doivent être définies dans l'implémentation du calcul des déséquilibres. Le code doit être lisible et facilement analysable au regard de la formule (5).**
- **Le premier test doit donner l'affichage final suivant :**

```
----------------------
n°181 m=10 c=10 g=10
n°181 m=20 c=8 g=9.0
n°181 m=30 c=6 g=8.0
n°181 m=40 c=4 g=7.0
n°181 m=50 c=2 g=6.0 !
n°181 m=60 c=0 g=5.0
^^^^^^^^^^^^^^^^^^^^^^
```

- **Le second test doit donner l'affichage final suivant :**

```
----------------------
n°464 m=14 c=0 g=0
n°262 m=24 c=3 g=1.25
n°464 m=38 c=6 g=3.0 !
n°262 m=48 c=9 g=4.25 !
n°464 m=62 c=12 g=6.0 !
n°262 m=72 c=14 g=7.111111111111111
n°464 m=86 c=11 g=7.744186046511628 !
n°262 m=96 c=13 g=8.291666666666666
n°464 m=110 c=10 g=8.50909090909091
n°262 m=120 c=8 g=8.466666666666667
n°464 m=134 c=5 g=8.104477611940299 !
n°262 m=144 c=3 g=7.75 !
n°464 m=158 c=0 g=7.063291139240507
^^^^^^^^^^^^^^^^^^^^^^
```

3 Représentation graphique de la pile

Introduction

Tout comme pour la représentation graphique des planchettes, le module *VuePile* fournit la représentation graphique associée au modèle de la pile défini par le type abstrait *Pile*. Dans le même esprit, il aurait été possible de faire de même concernant la représentation textuelle de la pile (fonction *versChaine*), mais cette représentation n'est utile que pour des besoins de mise au point du code (débogage).

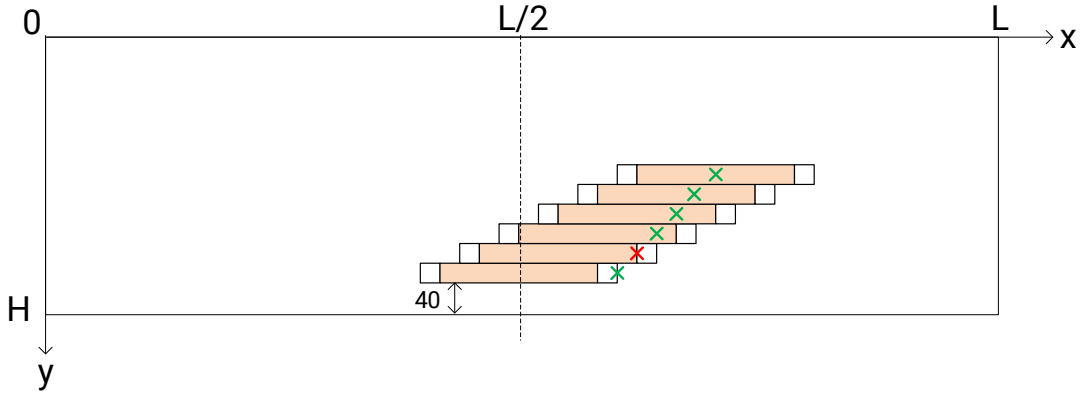

FIGURE 1 – Représentation graphique de la pile

La figure 1 illustre l'agencement de la représentation graphique de la pile, où *L* et *H* sont respectivement la largeur et la hauteur de la zone de dessin (ou « toile ») de la fenêtre de l'application. Le bas de la pile (premier empilement) est positionné en bas et au milieu de zone de dessin, avec une zone de 40 pixels de haut réservée pour l'affichage de boutons (étape 5). Chaque empilement précise la position de son centre de gravité par le symbole $\times$. Ce symbole passe du vert au rouge pour un empilement en déséquilibre.

Spécification

Le module *VuePile* ne contient qu'une fonction définie comme suit :

dessine(fenetre, pile) : dessine la *pile* à l'intérieur de la toile de la *fenetre*. Chaque empilement affichera sa planchette de base, ainsi que son centre de gravité symbolisé une croix verte si l'empilement est en équilibre ou rouge sinon.

Travail à faire

Étape 3.3

- Compléter le fichier « VuePile.py ». La fonction *dessine* devra définir différentes variables locales afin de rendre le code lisible. Bien entendu, la fonction *dessine* du module *VuePlanchette* est à utiliser dans cette fonction.
- Tester ce code en exécutant les modules « TestVuePile1.py » et « TestVue-

Pile2.py », lesquels reprennent les piles des modules « TestEquilibre1.py » et « TestEquilibre2.py »

Bilan :

- **la notion de modèle et de vue a été mise en œuvre sur un exemple complet… et complexe ;**
- **ce principe va être appliqué dans l'étape 4 suivante à l'identique pour implémenter les pioches des joueurs.**