Laboratoire 1

Objectifs (4h)

Au terme de ce laboratoire, vous serez capable de :

→ Réaliser une phase préparatoire, en ce compris un plan de test ;
→ Utiliser la phase préparatoire d'un[e] pair[e] pour programmer une solution ;
→ Utiliser un plan de test afin de valider sa solution.

Introduction – Implémentation d'une phase préparatoire (1h)

Avec votre responsable de laboratoire, parcourez la phase préparatoire proposée sur HELMo LEARN. Votre responsable de laboratoire insiste sur l'importance de la phase de test pour valider le plus exhaustivement possible une fonction. La phase préparatoire ne contient pas l'implémentation de la fonction. C'est à vous de l'implémenter.

À vous ! Énoncé général et contrainte (3h)

Formez des groupes des groupes de deux étudiants. Dans le binôme, choisissez qui sera l'étudiant[1] A et l'étudiant B. Installez-vous l'un à côté de l'autre.

La suite de ce labo se concentre sur l'utilité de la **phase préparatoire.** Pour l'illustrer, nous soumettons à tous les binômes deux problèmes : l'énoncé A et l'énoncé B. Toutefois, vous devez respecter une contrainte : vous ne programmerez pas la solution de l'énoncé dont vous avez réalisé la phase préparatoire ! Au contraire, après avoir chacun réalisé la phase préparatoire de l'un des énoncés, vous **échangerez** de phase avec votre binôme ! Le schéma suivant résume la situation :

	QUOI, TEST ET COMMENT	CODAGE + VALIDATION
ÉTUDIANT A	Avec Énoncé A	Avec prépa Étudiant B
ÉTUDIANT B	Avec Énoncé B	Avec prépa Étudiant A

Dans la suite, si vous ne comprenez pas ce qu'à voulu dire votre binôme, formulez-lui des critiques constructives, posez-lui des questions pour qu'il puisse améliorer son travail.

Conseil :

1. Vous pouvez, par exemple, partager un dossier via *OneDrive* dans lequel vous stockez les deux phases préparatoires. Vous pouvez également vous échanger des versions sur papier.
2. Aucun dispositif n'est prévu pour vous empêcher de lire l'autre l'énoncé mais l'exercice perd de l'intérêt si vous le faites.

[1] Pour des raisons d'ergonomie de lecture, le masculin sera utilisé à titre épicène dans l'ensemble des énoncés de labo.

Phase 1 – Lecture de l'énoncé et description du QUOI

Durée prévue : 30 min

1. Lisez attentivement l'énoncé.
2. Sur base de l'énoncé réalisez la phase « QUOI » de la phase préparatoire.
3. Quand vous avez fini, faites lire votre phase préparatoire à votre binôme. En ne se basant que sur votre travail, il formule toutes les questions de clarifications qui lui semblent nécessaires. Exemple de question : qu'elle est la valeur maximale de telle variable ? Qu'est-ce qui se passe si mon tableau contient telle valeur, etc.
4. Sur bases de ses remarques, complétez votre travail.

Phase 2 – Formaliser les tests

Durée prévue : 20 minutes

→ Complétez la partie 2 « Formalisation des tests » de **votre** phase préparatoire.

→ *Faites attention* : il n'y a que grâce à vous et les tests que vous lui fournissez que votre binôme pourra identifier des erreurs potentielles !

Phase 3 – Imaginer un algorithme et description du COMMENT

Durée prévue : 1h

1. Commencez par faire un dessin de votre problème. Aidez-vous des représentations qui sont données dans chacun des énoncés.
2. Identifiez si vous avez besoin d'une boucle (ce devrait être le cas).
3. Toutes les boucles ressemblent au schéma suivant :
   ```
   INITIALISATION
   while(GARDIEN) {
      CORPS DE LA BOUCLE
   }
   FIN
   ```
   Répertoriez textuellement ou en « pseudocode » ou encore sous forme d'organigramme, dans l'ordre, les étapes principales qui correspondent à `INITIALISATION`, `GARDIEN`, `CORPS DE LA BOUCLE`, et `FIN`. La phase préparatoire de l'exercice précédent vous donne un exemple.
   Attention : personne ne vous demande d'écrire du code Java à cette étape !
4. Faites en sorte d'assurer à votre binôme que vous n'êtes pas en train de lui faire coder une boucle infinie.

Phase 4 – Codage et validation

Durée prévue : 30 min

- → Échangez votre phase préparatoire avec celle de votre binôme.
- → Sur base de son travail préparatif, implémentez la solution.
- → Validez votre implémentation à l'aide des tests listé dans la phase préparatoire que vous utilisez.
- → Pendant que vous utilisez sa phase préparatoire, annotez-la avec des remarques :
    - Qu'avez-vous trouvé particulièrement utile dans ses explications ?
    - Qu'est-ce qui vous a semblé un plus obscur ?

Phase 5 – Retour réflexif sur ce qu'il vous était demandé

Durée prévue 20 min

- → Prenez maintenant connaissance de l'énoncé de votre binôme (celui dont vous venez de coder la solution). Imaginiez-vous l'énoncé rédigé de cette manière ?
- → Partagez avec votre binôme les notes que vous avez prises jusqu'à maintenant au sujet de sa phase préparatoire.
- → De cette discussion, essayer de distinguer :
    - Quels sont vos points forts, que devriez-vous retravailler ?
    - Où avez-vous perdu le plus de temps ? Comment y remédier à l'avenir ?

Liens cliquables vers les énoncés

Énoncé A

Énoncé B

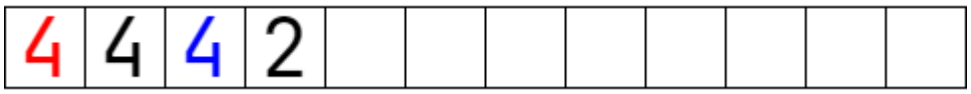
Énoncé A[2]

Le jeu 2048 est relativement populaire. Il consiste a essayer de fusionner des cases dans une grille contenant un des nombres qui sont des puissances de 2 jusqu'à obtenir 2048.

Intéressons-nous à une variante à **1 dimension**, que nous appellerons le 128. Dans notre problème, nous avons un tableau qui contient des puissances de 2 ou des cases vides (qui seront représentées par des zéros). Par exemple :

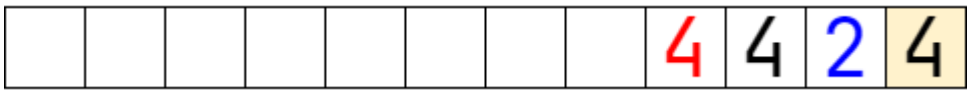
La fonction que nous voudrions réaliser va aussi recevoir un booléen qui est **vrai** si les blocs doivent être déplacés tous vers la gauche, et **faux** sinon.

Quand on déplace des blocs vers la gauche, tous les blocs sont déplacés vers ce côté et deux blocs de mêmes valeurs consécutifs sont fusionnés pour créer un bloc dont la valeur est doublée. Voici le résultat si les blocs sont déplacés vers la gauche :



Les chiffres colorés dans l'exemple de départs sont colorés identiquement dans le résultat.

Quand on déplace des blocs vers la droite, tous les blocs sont déplacés vers ce côté et deux blocs de mêmes valeurs consécutifs sont fusionnés pour créer un bloc dont la valeur est doublée. Voici le résultat si les blocs sont déplacés vers la droite :



Dans cet exemple, le fond jaune de la case à l'extrême droite représente les cases qui possèdent le même fond dans l'exemple.

Si une case apparait avec une valeur par fusion lors du déplacement, elle ne peut pas être fusionnée à nouveau. Par exemples, les 4 rouges n'ont pas été fusionnés avec le 4 noir qui leur est contigu.

On demande d'implémenter cette fonction qui modifie le tableau comme décrit ci-dessus. La classe contenant la fonction s'appellera `Jeu128`, sa classe de test `Jeu128Tests`.

Pour aller plus loin – à ne tenter que si vous avez le temps

Quand cela fonctionne avec le tableau 1D, vous pouvez tenter de faire la même chose dans un tableau en 2D. Les quatre mouvements disponibles sont `Haut`, `Bas`, `Gauche` et `Droite`. Pour reproduire le comportement de 2048, vous pouvez introduire un 2 ou un 4 dans une position aléatoire non occupée dans votre tableau.

[2] Inspiré du jeu 2048 (https://fr.wikipedia.org/wiki/2048_(jeu_vidéo))

Énoncé B[3]

Imaginez une planche en équilibre sur laquelle on installe des boules. Si on fait pencher la planche vers la gauche, les boules vont rouler vers la gauche. Si on l'a fait pencher vers la droite, les boules rouleront vers la droite (les boules ne peuvent pas tomber de la planche). Si maintenant, on fixe aussi des blocs sur cette planche et qu'on la fait pencher, les blocs qui sont sur le chemin de boules vont la bloquer. Une boule immobilisée par un bloc pourrait, à son tour, immobiliser une autre boule. Le mieux est de faire un dessin :

Dans votre programme, on représentera la planche comme un tableau de caractères. Le caractère `'O'` (la lettre O majuscule) représente une boule, le caractère `'#'` un bloc. Le caractère `' '` (espace) représente une position sur la planche qui n'est occupée ni par une boule, ni par un bloc.

La fonction que nous voudrions réaliser va recevoir un booléen qui est **vrai** si la planche penche à gauche, et **faux** sinon.

Voici le résultat si la planche penche vers la gauche :

On voit que la boule de gauche est déplacée à l'extrême gauche de la planche et que les autres ont été retenues par le bloc. La boule la plus à droite a été retenue par la boule qui est à sa gauche.

Voici le résultat si la planche penche vers la droite :

On demande d'implémenter cette fonction qui modifie un tableau de caractères comme décrit ci-dessus. La classe contenant la fonction s'appellera `PierreQuiRoule`, sa classe de test `PierreQuiRouleTests`.

Pour aller plus loin – à ne tenter que si vous avez le temps

Quand cela fonctionne avec le tableau 1D, vous pouvez tenter de faire la même chose dans un tableau en 2D. Les quatre mouvements disponibles sont `Haut`, `Bas`, `Gauche` et `Droite`. Dans un tel tableau 2D, si on répète des séquences des quatre mouvements `Haut`→ `Gauche` → `Bas` →`Droite` (dans cet ordre), au bout de combien de séquences les positions des boules dans le tableau se stabilisent-elles ?

[3] Inspiré de https://adventofcode.com/2023/day/14