Introduction à la Programmation **Java** (IP1-Java)
Examen – **Durée : 3 heures**

Université Paris-Diderot – Lundi 10 Décembre 2018

- Aucun document ni aucune machine ne sont autorisés. Les téléphones doivent être rangés.
- Les exercices sont tous indépendants.
- **Attention** : Indiquez au début de votre copie quel langage de programmation vous utiliserez dans le restant de votre copie. Il n'est pas possible de changer de langage une fois celui-ci choisi. Pour rappel, les filières INFO, MATHS-INFO et MIASHS Linguistique doivent composer en JAVA mais si vous voulez composer en PYTHON, merci de nous l'indiquer.
- Une réponse peut utiliser les réponses attendues à une question précédente (même si elle est non traitée).
- Les fragments de code doivent être correctement indentés.

Exercice 1

1. *Décrire le contenu des tableaux* t1 *et* t2 *après exécution du programme suivant.*

```
public class Test{
  public static int[] func(int[][] t){
    int [] tab=new int[t.length];
    for(int i=0;i<tab.length;i=i+1){
      tab[i]=0;
    }
    for(int i=0;i<t.length;i=i+1){
      for(int j=0;j<t[i].length;j=j+1){
          if(j<i) {
            tab[i]=tab[i]+t[i][j];
          }
      }
    }
    return tab;
  }

  public static void main(String[] args){
    int [][]t1 = {{1,2},{1,2,3},{1,2,3,4}};
    int []t2 = func(t1);
  }
}
```

2. *Donner la valeur de la variable* a *après exécution du programme suivant.*

```
public class Test2{

  public static int f(int x){
    int y=0;
    if(x%4==0) {
      y = -x;
    } else {
      y = x;
    }
    return y;
  }

  public static int g(int c){
    while(f(c)>0){
      c=c+1;
    }
    return c;
  }

  public static void main(String[] args){
```

```
    int a = g(5);
  }
}
```

□

Exercice 2 *Dans cet exercice, nous considérons un ensemble de tiroirs avec une certaine capacité (le nombre d'éléments maximal que l'on peut mettre dedans). Nous représentons cet ensemble par un tableau d'entiers où l'indice représente le numéro du tiroir et la valeur le nombre d'éléments dans le tiroir. Ainsi le tableau* {1,0,3} *représente un ensemble de trois tiroirs où le tiroir 0 a 1 élément, le tiroir 1 n'a pas d'élément et le tiroir 2 a 3 éléments.*

1. *Écrire une fonction* retire *qui prend en arguments un tableau d'entiers représentant un ensemble de tiroirs, un numéro de tiroirs* num *et qui retire un élément du tiroir* num *et qui renvoie* true *si cela est possible (le tiroir existe et il a au moins un élément) et* false *sinon. Dans le cas où ce n'est pas possible, les tiroirs restent inchangés.* **Exemple :** *Si* T={1,0,2} *alors* retire (T,0) *renvoie* true *et après l'appel* T *vaut* {0,0,2}*, en revanche* retire (T,1) *renvoie* false *et* T *reste inchangé.*
2. *Écrire une fonction* plein *qui prend en arguments un tableau d'entiers représentant un ensemble de tiroirs et un entier* cmax *représentant la capacité maximale pour les tiroirs et qui renvoie* true *si tous les tiroirs sont pleins (c'est-à-dire que chaque tiroir a un nombre d'éléments supérieur ou égal à* cmax*) et* false *sinon.* **Exemple :** *Si* T1={1,0,2} *alors* plein (T1,2) *renvoie* false *, si* T2={3,3,3} *alors* plein (T2,3) *renvoie* true *et si* T3={3,4,5} *alors* plein (T3,3) *renvoie* true*.*
3. *Écrire une fonction* tiroirsLibres *qui prend en arguments un tableau d'entiers représentant un ensemble de tiroirs et un entier* cmax *représentant la capacité maximale pour les tiroirs et qui renvoie le tableau contenant les numéros des tiroirs dont le nombre d'éléments est strictement inférieur à* cmax*.* **Exemple :** *Si* T1={1,0,2} *alors* tiroirsLibres (T1,2) *renvoie* {0,1}*, si* T2={3,3,3} *alors* tiroirsLibres (T2,3) *renvoie* {} *et si* T3={5,4,6,2,3} *alors* tiroirsLibres (T3,5) *renvoie* {1,3,4}*.*
4. *On veut ajouter un élément dans un tiroir. Pour cela, on donne un numéro de tiroir, si il y a de la place dedans on rajoute l'élément sinon on rajoute au premier tiroir libre qui se trouve après et si il n'y pas de place dans tous les tiroirs après, on revient au tiroir 0 et on cherche dans celui-ci et dans ceux après. Écrire une fonction* ajouteElem *qui prend en arguments un tableau d'entiers représentant un ensemble de tiroirs, un entier* cmax *représentant la capacité maximale pour les tiroirs et un numéro de tiroir où l'on veut ajouter l'élément et qui renvoie le tableau correspondant au nouvel ensemble avec l'élément ajouté (en suivant les règles d'ajout données ci-dessus). Si l'on ne peut pas ajouter l'élément, alors la fonction renvoie le tableau inchangé. On suppose que le numéro de tiroir est correct.* **Exemple :** *Si* T1={1,0,2} *alors* ajouteElem(T1,2,1) *renvoie* {1,1,2}*, si* T2={2,2,1} *alors* ajouteElem(T2,2,1) *renvoie* {2,2,2}*, si* T3={3,2,3,1,3} *alors* ajouteElem(T3,3,4) *renvoie* {3,3,3,1,3} *et si* T4={3,3,3,3} *alors* ajouteElem(T4,3,1) *renvoie* {3,3,3,3}*, .*
5. *On veut maintenant ajouter un ensemble d'éléments aux tiroirs en suivant les règles de la question précédente. Pour ce faire, les éléments à ajouter seront donnés par un tableau où dans chaque case, on note le numéro du tiroir où l'on souhaite ajouter l'élément. Par exemple,* {1,0,2,2} *voudra dire que l'on souhaite ajouter un élément au tiroir 1 puis un élément au tiroir 0, puis un élément au tiroir 2 et un dernier élément au tiroir 2. Écrire une fonction* ajoutePlusieursElem *qui prend en arguments un tableau d'entiers représentant un ensemble de tiroirs, un entier* cmax *représentant la capacité maximale pour les tiroirs et un tableau stockant les numéros de tiroirs où l'on veut ajouter des éléments et qui renvoie le tableau correspondant au nouvel ensemble avec les éléments ajoutés (en suivant les règles d'ajout données ci-dessus). Si l'on ne peut pas ajouter tous les éléments, le tableau renvoyé contiendra l'ensemble où l'on a ajouté un nombre maximal d'éléments avant que les tiroirs ne soient pleins. On suppose que les numéros de tiroirs sont corrects.* **Exemple :** *Si* T1={1,0,2} *alors* ajoutePlusieursElem (T1,2,{1,0,1}) *renvoie* {2,2,2} *et si* T2={2,2,1} *alors* ajoutePlusieursElem (T2,2,{0,1,0}) *renvoie* {2,2,2}*.*

□

Exercice 3 *Dans cet exercice, on considère des dictionnaires qui sont représentés par des tableaux de chaînes de caractères où chaque chaîne ne contient que des lettres minuscules (pas de symbole ni de chiffre). En* JAVA*, pour les questions suivantes, vous pouvez utiliser par exemple :*

- *«* String characterAtPos (String s, int i) *» qui renvoie la chaîne de longueur* 1 *à la position* i *de* s *(on rappelle que la première position est la position* 0*).*
- *«* int stringLength (String s) *» qui renvoie la longueur de la chaîne* s*.*
- *«* boolean stringEquals (String s1, String s2) *» qui renvoie* true *si les deux chaînes* s1 *et* s2 *sont égales et* false *sinon.*

1. *Écrire une fonction* tailleMotPlusLong *qui étant donné un tableau de chaînes de caractères représentant un dictionnaire renvoie la taille du mot le plus long.* **Exemple :** *Si* D={"monde","le","petit"} *alors* tailleMotPlusLong(D) *renvoie* 5*.*
2. *Écrire une fonction* motsPremiereLettre *qui étant donnés un tableau de chaînes de caractères représentant un dictionnaire et une chaîne de caractères avec un seul caractère* ch *renvoie le tableau contenant tous les mots du*

dictionnaire dont la première lettre est ch. **Exemple :** *Si* D={"monde","le","petit","merci"} *alors* motsPremiereLettre (D,"m") *renvoie* {"monde","merci"}.

3. *Écrire une fonction* effaceEspace *qui prend en entrée une chaîne de caractères avec que des lettres minuscules et des espaces et qui renvoie le tableau contenant les mots contenus dans la chaîne sans les espaces. On supposera qu'entre deux lettres il y a au plus un espace.* **Exemple :** effaceEspace("le monde est petit") *renvoie* {"le","monde","est","petit"}. *Si un mot apparaît plusieurs fois dans la chaîne, il apparaitra plusieurs fois dans le tableau renvoyé.*
4. *Écrire une fonction* estCorrect *qui prend en entrée une chaîne de caractères avec que des lettres minuscules et des espaces et un tableau de chaînes de caractères représentant un dictionnaire et qui renvoie* true *si tous les mots de la chaîne sont dans le dictionnaire et* false *sinon.* **Exemple :** *Si* D={"monde","le","petit","merci"} *alors* estCorrect("le monde est petit",D) *renvoie* false *car le mot* "est" *n'apparaît pas dans le dictionnaire.*
5. *Écrire une fonction (procédure)* corrige *qui prend en entrée une chaîne de caractères avec que des lettres minuscules et des espaces et un tableau de chaînes de caractères représentant un dictionnaire et qui affiche la chaîne de caractères correspondant à la chaîne originale où tous les mots n'appartenant pas au dictionnaire ont été effacés.* **Exemple :** *Si* D={"monde","le","petit","merci"} *alors* corrige("le monde est petit",D) *affiche* le monde petit.

□

Exercice 4 *Dans cet exercice, on suppose qu'un ensemble de bulletins est un tableau de tableaux d'entiers dans lequel à la case d'indice i se trouve le tableau représentant les notes que l'étudiant i a eu dans chaque matière. On suppose que tous les étudiants ont le même nombre de matières et ont une note entre 0 et 20 pour chaque matière.*

1. *Écrire une fonction* moyenneEtu *qui prend comme arguments un ensemble de bulletins (représenté sous forme de tableau de tableaux d'entiers) et un numéro d'étudiant et renvoie la moyenne entière de cet étudiant.*
2. *Écrire une fonction* passe *qui prend comme argument un ensemble de bulletins (représenté sous forme de tableau de tableaux d'entiers) et renvoie le nombre d'étudiants qui ont leur année, c'est à dire dont la moyenne de leurs notes est supérieure ou égale à 10.*
3. *Écrire une fonction* meilleurs *qui prend comme arguments un ensemble de bulletins (représenté sous forme de tableau de tableaux d'entiers) et renvoie un tableau dont les éléments sont les numéros des étudiants ayant eu la meilleure moyenne.*

□

Exercice 5 *Le but de cet exercice est de programmer un tetris simplifié. À chaque tour, l'ordinateur propose une pièce au joueur et le joueur dit ensuite dans quelles colonnes il souhaite que la pièce tombe. Une fois la pièce tombée, lorsqu'une (ou plusieurs) ligne est complète en bas, elle disparaît et le jeu recommence tant que l'utilisateur insère les pièces où il peut. Dans ce jeu, on représentera une grille de jeu à M lignes et N colonnes par un tableau de M tableaux de N entiers. Les lignes sont numérotés de 0 à M-1 et les colonnes de 0 à N-1. Ce tableau ne contiendra que des 0 pour les endroits libres et des 1 pour les endroits occupés par une pièce. La ligne 0 sera celle d'en haut et la ligne M-1 celle d'en bas et la colonne 0 sera celle de gauche et la N-1 celle de droite. Il y a quatre types de pièces numérotées de 1 à 4 dont la forme est donnée par la figure ci-dessous :*

Pièce 1 *Pièce 2* *Pièce 3* *Pièce 4*

Quand l'ordinateur propose un numéro de pièce, le joueur donne le numéro de la colonne la plus à gauche où la pièce doit tomber et celle-ci tombe jusqu'à toucher une autre pièce ou à arriver au bas de la grille. Par exemple, si le jeu est dans la Configuration 1 ci-dessous, que la pièce sélectionnée est la pièce 3 et le joueur donne le numéro de colonne 2, la pièce tombera et le jeu sera dans la configuration 2. Depuis la configuration 2, si le joueur propose d'insérer une pièce dans la colonne 3 qui est pleine il perd, mais aussi si il propose d'insérer une pièce 2 dans la colonne 2 car pour rentrer celle-ci a besoin que les colonnes 2 et 3 soient libres. Si en revanche, depuis la configuration 2, l'ordinateur propose la pièce 4 et le joueur dit de la mettre dans la colonne 0, le jeu arrive dans la configuration 3.

0	0	0	0	0
0	0	0	0	0
0	0	0	1	0

Configuration 1

0	0	0	1	0
0	0	1	1	0
0	0	0	1	0

Configuration 2

0	0	0	1	0
1	1	1	1	0
1	1	0	1	0

Configuration 3

Une fois que la pièce est tombée, le jeu efface les lignes du bas complètes avec que des 1 avant de proposer une autre pièce au joueur. Ainsi, si le jeu est dans la configuration 4 ci-dessous, il passe à la configuration 5.

1	0	0	1	0
1	1	1	1	0
1	1	1	1	1
1	1	1	1	1

Configuration 4

0	0	0	0	0
0	0	0	0	0
1	0	0	1	0
1	1	1	1	0

Configuration Gagnante 5

1. *Écrire une fonction* creerConfiguration *qui prend comme arguments deux entiers* M *et* N *et renvoie un jeu avec* M *lignes et* N *colonnes où toutes les cases sont libres.*
2. *Écrire une fonction (procédure)* afficheConfiguration *qui prend comme argument une grille de jeu et affiche son contenu sur le terminal ligne par ligne en séparant chaque case par un espace et en affichant le caractère moins (-) pour les cases libres et un O pour les cases occupées.* **Exemple :** *Pour la configuration 2, la fonction* afficheGrille *affichera :*

   \- \- \- O \-
   \- \- O O \-
   \- \- \- O \-
3. *Écrire une fonction* insertion *qui prend comme arguments une grille correspondant à une configuration du jeu, un numéro de pièce et un numéro de colonne et qui renvoie* true *si la pièce peut être insérée (la colonne donnée étant celle la plus à gauche de la pièce) et* false *sinon et de plus elle modifie la grille pour insérer la pièce si elle peut l'insérer et sinon elle ne change pas la grille. Par exemple, si* gr *est le tableau de tableaux d'entiers correspondant à la configuration 1 ci-dessus, alors* insertion(gr,3,2) *renvoie* true *et la grille est changée pour correspondre à la configuration 2. Attention, il faut vérifier que la pièce puisse être insérée en entier et il faut donc chercher la ligne la plus basse telle que toutes les lignes au-dessus permettent de faire passer la pièce.*
4. *Écrire une fonction* choixPiece *qui ne prend pas d'argument et renvoie un numéro de pièce tiré au hasard entre 1 et 4 inclus. On pourra se servir d'une fonction* int alea(int a, int b) *qui renvoie un entier au hasard supérieur ou égal à* a *et strictement inférieur à* b.
5. *Écrire une fonction* colonne *qui prend comme arguments une grille et un numéro de pièce et qui affiche la grille (grâce à la fonction de la question 2), puis affiche le numéro de pièce et demande à l'utilisateur un numéro de colonnes et renvoie le numéro tapé au clavier par l'utilisateur. On pourra se servir d'une fonction* int getInt() *qui attend que l'utilisateur ait tapé un entier au clavier et le renvoie.*
6. *Écrire une fonction* estSimple *qui prend en argument une grille représentant une configuration du jeu et renvoie* true *si la ligne la plus basse ne contient pas que des 1 sinon elle renvoie* false .
7. *Écrire une fonction (procédure)* supprimeLigne *qui prend en arguments une grille représentant une configuration du jeu et supprime la dernière ligne, baisse toutes les autres lignes d'un cran et insère en haut une ligne avec que des 0. Cette fonction ne renvoie rien.*
8. *Écrire une fonction* simplifie *qui prend en argument une grille représentant une configuration du jeu et tant que celle-ci a sa dernière ligne en bas avec que des 1, enlève cette dernière ligne, baisse les autres et met une ligne remplie de 0 en haut. Cette fonction renvoie le nombre de lignes supprimées.*
9. *Écrire une fonction (procédure)* jouer *qui prend en arguments un nombre lignes M et un nombre de colonnes N et qui fait jouer depuis le début sur une grille à M lignes et N colonnes le joueur de la façon suivante : en boucle le programme propose une pièce au hasard, puis demande à l'utilisateur de rentrer un numéro de colonnes pour insérer la pièce (grâce à la fonction* colonne*), si la pièce peut être insérée, on insère la pièce puis on supprime les lignes en bas avec que des 1 et on recommence. Si l'utilisateur rentre un numéro de colonnes où la pièce ne peut pas être insérée, le jeu s'arrête et le programme affiche le nombre de points obtenus qui correspond au nombres de lignes supprimées depuis le début du jeu. Cette fonction ne renvoie rien.*

□