Introduction à la Programmation **Java** (IP1-Java)
Examen – **Durée : 2 heures**

Université de Paris– Lundi 7 Décembre 2020

- Aucun document ni aucune machine ne sont autorisés. Les téléphones doivent être rangés.
- Les exercices sont tous indépendants.
- 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.
- Pour certains exercices, vous pouvez utiliser par exemple :
  - « String charAtPos (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.

  Vous pouvez cependant utiliser tout équivalent JAVA (Par exemple, `s.length()` à la place de `stringLength(s)`).
- Le sujet est long et il est possible qu'il soit noté sur plus que 20.

Exercice 1

1. *Donnez la valeur de la variable* m *après exécution du programme suivant :*

```
public class Test1{
  public static int g(int x){
    int res=2;
    int y=1;
    while(y<=x && y%4!=0){
      res=res*2;
      y=y+1;
    }
    return res;
  }

  public static String[] f(String s){
    String[] stab=new String[s.length()];
    for(int i=stab.length-1;i>=0;i=i-1){
      stab[i]="";
        for(int j=0;j<i;j=j+1){
          stab[i]=stab[i]+s;
        }
     }
     return stab;
  }

  public static void main(String[] args){
    String st="BOUM";
    String[] t=f(st);
    int m=g(t[2].length());
  }
}
```

2. *Donnez le contenu du tableau* tt *et de ses sous-tableaux après exécution du programme suivant :*

```
public class Test2{
  public static void main(String[] args){
    int[][] tt=new int[4][];
    for(int i=0;i<4;i=i+1){
      if(i%2==0){
        tt[i]=new int[i+1];
      }else{
        tt[i]=new int[i-1];
      }
```

```
        int p=0;
        for(int j=0;j<tt[i].length;j=j+1){
          tt[i][j]=p;
          p=p+1;
        }
      }
    }
}
```

On considère la fonction func *suivante.*

```
public static int[] func(int[] tab){
  for(int i=0;i<tab.length;i=i+1){
    int tmp=tab[i];
    tab[i]=tab[tab.length-1-i];
    tab[tab.length-1-i]=tmp;
  }
  return tab;
}
```

3. *Si l'on a* int [] t={10,5,3,12,7} *quelle forme a le tableau renvoyé par l'appel à* func(t) *?*
4. *Que fait la fonction* func *?*

□

Exercice 2

1. *Écrire une fonction* diviseurs *qui prend comme argument un entier supposé strictement positif et renvoie un tableau d'entiers contenant la liste de ses diviseurs. Par exemple,* diviseurs (10) *renverra le tableau* {1,2,5,10}.
2. *Écrire une fonction* diviseursCommuns *qui prend comme argument un tableau non vide d'entiers (supposés tous strictement positifs)* tab *et renvoie un tableau d'entiers contenant les diviseurs qui sont communs à tous les éléments de* tab. *Par exemple, si l'on a* int [] tab1={5,10} *alors* diviseursCommuns(tab1) *renverra le tableau* {1,5}, *et si* int [] tab2={2,5,10} *alors* diviseursCommuns(tab2) *renverra le tableau* {1}.
3. *Écrire une fonction* pgcd *qui prend comme argument un tableau non vide d'entiers (supposés tous strictement positifs)* tab *et renvoie le plus grand diviseur commun à tous les éléments de* tab. *Par exemple, si l'on a* int [] tab1={5,10} *alors* pgcd(tab1) *renverra* 5, *et si* int [] tab2={2,5,10} *alors* pgcd(tab2) *renverra* 1.
4. *Écrire une procédure* plusPetitSommeDiv *qui prend en argument un entier* n *supposé strictement positif et qui affiche le plus petit entier tel que la somme de ces diviseurs soit supérieure ou égale à* n. *Par exemple,* plusPetitSommeDiv (10) *affichera* 6 *car les diviseurs de* 6 *sont* 1, 2, 3 *et* 6 *et leur somme vaut* 12 *(qui est supérieure ou égale à* 10*) et pour tout entier plus petit que* 6 *la somme des diviseurs est strictement inférieure à* 10.

□

Exercice 3 *Les numéros d'étudiant.e de l'Université de Paris sont des chaîne de caractères de longueur 10 commençant par "UP" et dont les 8 derniers caractères sont des chiffres compris entre 0 et 9. Par exemple la chaîne de caractères* "UP50218932" *respecte ce format.*

1. *Écrire une fonction* boolean checkNumero(String ch) *qui vérifie si la chaîne* ch *respecte le format des numéros d'étudiant.e de l'Université de Paris. Cette fonction renverra* true *si* ch *respecte le format et* false *sinon. Par exemple,* checkNumero("UP50218932") *renverra* true, checkNumero("OP50218932") *renverra* false *et* checkNumero("UP502189") *renverra* false.
2. *Écrire une fonction* boolean checkTab(String []tab) *qui vérifie si un tableau* `tab` *de chaînes de caractères ne contient que des chaînes de caractères respectant le format des numéros d'étudiant.e. et renvoie* true *si c'est le cas et* false *sinon.*
3. *Écrire une fonction* boolean checkDiff(String []tab) *qui vérifie qu'un tableau* tab *de chaînes de caractères ne contient pas deux fois la même chaîne. Elle renvoie* true *si on n'a pas deux fois la même chaîne dans le tableau et* false *sinon.*
4. *Nous voulons maintenant comparer deux numéros d'étudiant.e. Écrire une fonction* boolean plusGrand(String ch1, String ch2) *qui prend comme arguments deux chaînes de caractères et qui renvoie* true *si les deux chaînes sont bien au format de numéro d'étudiants et le numéro de la première chaîne et strictement plus grand que le numéro de la deuxième. Dans les autres cas, cette fonction renvoie* false. *Par exemple* plusGrand("UP12009734","UP12009735") *renvoie* false *et* plusGrand("UP13009734","UP12009735") *renvoie* true.
5. *Écrire une fonction* boolean present(String [] tab, String st) *qui renvoie* true *si la chaîne* st *est présente dans le tableau* tab *et qui renvoie* false *sinon.*

6. *L'Université de Paris a décidé de donner accès à ses salles de sport par groupes limitées de n personnes pour éviter les encombrements. Écrire une fonction* String [] formeGroupe(String [] tab, int n) *qui prend en arguments un tableau* tab *de numéros d'étudiant.e et un entier* n *tels que si* $0 <$ n$<$ tab.length *alors cette fonction renvoie un nouveau tableau de taille* n *contenant* n *numéros d'étudiant.e.s tous différents pris au hasard dans* tab *et si* tab.length$\leq$ n *ou si* n≤ 0 *cette fonction renvoie une copie du tableau* tab. *On pourra se servir d'une fonction* randRange(int a, int b) *qui renvoie un entier tiré au hasard compris entre* a *inclus et* b *exclus.*

□

Exercice 4 *Vous êtes développeur junior chez Nitflex, une plate-forme en ligne de streaming de films, et vous souhaitez programmer un système permettant de gérer les votes des utilisateurs pour ces films. Pour faire cela, on suppose qu'un tableau de votes est un tableau à deux dimensions* int [][] vote *de taille n (le nombre d'utilisateurs dans le système). Chacun des tableaux internes est de taille d (le nombre de films dans le système). Le tableau se lit de la manière suivante :* vote[i][f] *contient l'avis de l'utilisateur* i *sur le film* f. *L'avis peut avoir trois valeurs entières :* 0 *si l'utilisateur n'a pas aimé le film,* 1 *si l'utilisateur l'a aimé ou* -1 *si l'utilisateur ne l'a pas vu.*

Par exemple, le tableau $\{\{0,0,0\},\{1,0,-1\}\}$ *correspond à un tableau de votes avec deux utilisateurs et trois films où le premier utilisateur a vu les trois films et n'en a aimé aucun et le deuxième utilisateur a vu les deux premiers films et pas le dernier et a aimé le premier film mais pas le deuxième.*

1. *Écrire une fonction* filmsVus(int [][] vote, int i) *qui prend en arguments un tableau de votes et un numéro d'utilisateur et qui renvoie le nombre de films vus par cet utilisateur (si il n'y a pas d'utilisateur avec ce numéro dans le tableau de votes, la fonction renvoie* -1*). Par exemple si l'on a* int [][] vo$=\{\{0,0,0\},\{1,0,-1\}\}$ *alors* filmsVus(vo, 1) *renverra* 2.
2. *Écrire une fonction* plusAime *qui prend en argument un tableau de votes et renvoie le numéro du film le plus aimé (si ce tableau est vide, la fonction renvoie* -1*). Si il y a deux films qui ont été les plus aimés, la fonction renvoie le numéro de l'un des deux. Par exemple si l'on a* int [][] vo$=\{\{0,0,0\},\{1,0,-1\},\{1,1,-1\}\}$ *alors* plusAime(vo) *renverra* 0 *car le film* 0 *a été aimé par deux utilisateurs.*
3. *Écrire une fonction* filmsAimes *qui prend en arguments un tableau de votes et un numéro d'utilisateur et qui renvoie un tableau d'entiers contenant les numéros des films que l'utilisateur a aimés (si il n'y a pas d'utilisateur avec ce numéro dans le tableau de votes, la fonction renvoie le tableau vide). Par exemple, si l'on a* int [][] vo $=\{\{0,0,0\},\{1,0,-1\},\{1,0,1\}\}$ *alors* filmsAimes(vo, 2) *renverra le tableau* $\{0,2\}$ *car l'utilisateur* 2 *a aimé les films* 0 *et* 2.
4. *Écrire une fonction* filmsCommuns *qui prend en arguments un tableau de votes, un numéro d'utilisateur* i *et un numéro d'utilisateur* j *supposés corrects et qui renvoie un tableau d'entiers contenant les numéros des films que les deux utilisateurs ont aimés. Par exemple, si l'on a* int [][] vo$=\{\{0,0,0\},\{1,0,-1\},\{1,0,1\}\}$ *alors* filmCommuns(vo, 1,2) *renverra le tableau* $\{0\}$ *car les utilisateurs* 1 *et* 2 *n'ont tous les deux aimé qu'un seul film qui est le numéro* 0.
5. *Écrire une fonction* ajouteFilm *qui prend en arguments un tableau de votes et renvoie un nouveau tableau de votes auquel on a ajouté un film que l'on suppose que personne n'a vu. Par exemple, si l'on a* int [][] vo$=\{\{0,0,0\},\{1,0,-1\},\{1,0,1\}\}$ *alors* ajouteFilm(vo) *renverra le tableau de votes* $\{\{0,0,0,-1\},\{1,0,-1,-1\},\{1,0,1,-1\}\}$.

On veut maintenant associer des noms aux utilisateurs grâce à un tableau de chaînes de caractères String [] noms *où* noms[i] *contient le nom de l'utilisateur* i,

6. *Écrire une procédure* afficheVote *qui prend en arguments un tableau de votes et un tableau de noms (on suppose que ces deux tableaux ont la même longueur) et qui affiche par ligne, le nom de l'utilisateur et les films qu'il a aimé (si cet utilisateur a aimé au moins un film). Par exemple, si l'on a* int [][] vo$=\{\{1,0,0\},\{0,0,0\},\{1,1,-1\}\}$ *et* String [] no$=\{$"Alice","Bob","Layla"$\}$ *alors* afficheVote(vo,no) *affichera :*

```
Alice aime les films : 0
Layla aime les films : 0, 1
```

7. *On veut pouvoir supprimer un utilisateur et mettre à jour le tableau de votes. Écrire une fonction* supprimeVote *qui prend en arguments un tableau de votes, un tableau de noms et un nom d'utilisateur* no *et qui renvoie un tableau de votes où la ligne correspondant à l'utilisateur de nom* no *a été supprimée (l'ordre entre les autres utilisateurs restent le même). Si le nom* no *n'est pas dans le tableau de noms, la fonction renvoie le tableau de votes inchangè. Par exemple, si l'on a* int [][] vo$=\{\{1,0,0\},\{0,0,0\},\{1,1,-1\}\}$ *et* String [] no$=\{$"Alice","Bob","Layla"$\}$ *alors* supprimeVote(vo,no,"Bob") *renverra le tableau de votes* $\{\{1,0,0\},\{1,1,-1\}\}$.

□