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

Université Paris-Diderot – Samedi 9 Novembre 2019

- 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.
- Dans les énoncés, on propose parfois une liste de fonctions et procédures utilisables. Vous pouvez cependant utiliser tout équivalent JAVA (Par exemple, `s.length()` à la place de `stringLength(s)`). Attention cependant à ne pas utiliser des fonctions de la bibliothèque standard qui n'auraient pas été abordées en cours.
- Le sujet est long et il est possible qu'il soit noté sur plus que 20.

Exercice 1

1. *Qu'écrire à la place de* A, B, C, D *et* E *pour que la fonction «* func *» suivante soit correcte.*

```
public static A func (B x, C y, D st) {
  E res="";
  for(int i=0;i<=x;i=i+1){
     for(int j=0;j<=y;j=j+1){
        res=res+st;
      }
      res=res+"\n";
  }
  return res;
}
```

2. *Quelle est la valeur* st *à la fin de l'exécution des instructions suivantes ?*

```
String a="ABCDEF";
String st=a;
for (int i=1;i<=a.length()-3;i=i+1){
  st="+"+st+"+";
}
for (int j=0;j<=4;j=j+1){
  if(j%2 == 0){
    st="#"+st;
  } else{
    st=st+"#";
  }
}
```

3. *On considère la fonction* foo *dont la définition est donnée ci-dessous. Que renvoie* foo(t) *si l'on a* int []t={2,3,0,1} *?*

```
public static int[] foo(int[] x){

  int[]tab=new int[2*x.length];
  for(int i=0;i<x.length;i=i+1){
    tab[2*i]=2*x[i];
    tab[2*i+1]=x[i]-1;
  }
  int v=tab[tab.length-1];
  for(int j=tab.length-1;j>0;j=j-1){
    tab[j]=tab[j-1];
  }
  tab[0]=v;
  return tab;
```

```
}
```

□

Exercice 2

1. *Écrire une procédure* creneauxSimple *qui prend comme argument un entier* lo *et affiche deux lignes chacune de longueur deux fois* lo. *Chacune de ces lignes sera composée d'une alternance de* + *(symbole plus) et de* − *(symbole moins), la première commencera par un* + *et la deuxième par* −. *Par exemple,* creneauxSimple(4) *affiche :*
   +−+−+−+−
   −+−+−+−+
2. *Écrire une procédure* creneaux *qui prend comme arguments un tableau d'entiers et qui affiche deux lignes ayant comme nombre de caractères la somme des éléments dans le tableau. Chaque ligne sera composée d'une alternance d'un nombre de* + *(symbole plus) et d'un nombre de* − *(symbole moins), ces nombres correspondant aux entiers écrits dans le tableau. La première ligne commencera par des* + *et la deuxième par des* −. *Ainsi si le tableau donné en argument est* tab, *la première ligne affichera* tab[0] *fois* +, *puis* tab[1] *fois* − *puis* tab[2] *fois* +, *etc. Et la deuxième ligne affichera la même chose en inversant les* + *et les* −. *Par exemple, si l'on a* int [] tab={2,1,4}, *l'appel à* creneaux(tab) *affiche :*
   ++−++++
   −−+−−−−
3. *Écrire un programme* public static void main(String[] args) *qui utilise la fonction précédente et fait afficher :*
   +++++++
   −−−−−−−
   +−−−−−+
   −+++++−
   ++−−+−−
   −−++−++

□

Exercice 3 *Pour gérer les notes d'un examen, le professeur Tournesol a décidé de les stocker sous la forme d'un tableau d'entiers. Chaque copie est d'abord anonymisée, puis les copies sont numérotées de* 1 *à* n *où* n *est le nombre d'élèves. Nous retrouverons donc dans la case d'indice* i *du tableau les notes de la* $i+1$*-ème copie. Le professeur remplit le tableau au fur et à mesure et au début il y a des* -1 *dans les cases où il n'y a pas de note. Le professeur ne remplit pas forcément le tableau dans l'ordre.*

1. *Écrire une fonction* indNonRemp *qui prend en argument un tableau de notes et renvoie l'indice le plus petit d'une case non remplie. Cette fonction renvoie* -1 *si toutes les cases sont remplies. Par exemple, si l'on a* int [] notes ={18,15,−1,4,15}, *alors* indNonRemp(notes) *renverra* 2.
2. *Écrire une fonction* plein *qui prend en argument un tableau de notes et renvoie le booléen* true *si le tableau a toutes ses cases remplies et* false *sinon. Par exemple, si l'on a* int [] notes={18,15,−1,4,15}, *alors* plein(notes) *renverra* false.
3. *Écrire une fonction* nbrMoyenne *qui prend en argument un tableau de notes et renvoie le nombre d'étudiants qui ont une note supérieure ou égale à 10. Par exemple, si l'on a* int [] notes={18,15,−1,4,15}, *alors* nbrMoyenne(notes) *renverra* 3.
4. *Écrire une fonction* quiPasse *qui prend en argument un tableau de notes* notes *et renvoie un tableau de chaînes de caractère indiquant* "passe" *dans les cases d'indice* i *telles que* notes[i] *est entre 10 et 20 inclus,* "ajourne" *dans les cases d'indice* i *telles que* notes[i] *est entre 0 inclus et 10 exclu et* "erreur" *dans les autres cases. Par exemple, si l'on a* int [] notes={18,15,−1,4,15}, *alors* quiPasse(notes) *renverra* {"passe","passe","erreur","ajourne","passe"}.

□

Exercice 4 *En* JAVA, *pour les questions suivantes, 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.*

La base azotée d'un brin d'ADN (Acide DésoxyriboNucléique) peut être vue comme une chaîne de caractères ne contenant que les caractères : A, T, C et G (correspondant aux quatre nucléotides différents : l'adénine (A), la thymine (T), la cytosine (C) et la guanine (G)). Dans cet exercice, nous allons écrire des fonctions pour manipuler de telles bases azotées.

1. *Écrire une fonction* estBase *qui prend en argument une chaine de caractères et renvoie* true *si elle correspond à une base azotée (elle a au moins un caractère et tous ses caractères sont des A, des T, des C ou des G) et* false *sinon. Par exemple,* estBase("THACG") *renverra* false *et* estBase("TAACG") *renverra* true.

2. *Le complément d'une base azotée est une chaîne de caractères de même longueur où les A sont remplacés par des T, les T par des A, les C par des G et les G par des C. Écrire une fonction* complement *qui prend en argument une chaine de caractères, si cette chaine ne correspond pas à une base azotée alors la fonction renvoie la chaîne vide* "" *et sinon elle renvoie la chaîne correspondant au complément de la base azotée. Par exemple,* complement("THACG") *renverra* "" *et* complement("TAACG") *renverra* "ATTGC".

3. *Deux bases azotées peuvent être emboitées si l'une est le complément de l'autre. Écrire une fonction* semboite *qui prend en arguments deux chaînes de caractères et renvoie* true *si les deux chaînes sont des bases azotées et la deuxième est le complément de la première, dans tous les autres cas la fonction renvoie* false *. Par exemple,* semboite("THACG","TAACG") *renverra* false *et* semboite("ATTGC","TAACG") *renverra* true*.*

4. *On souhaite maintenant connaître le pourcentage de chaque nucléotide dans une base azotée. On écrit une fonction* composition *qui prend en argument une chaîne de caractères (que l'on supposera encodant une base azotée) et renvoie un tableau de quatre entiers où dans la première case on trouvera le pourcentage correspondant à la proportion entière du nombre de A dans la base, dans la deuxième case on trouvera le pourcentage correspondant à la proportion entière du nombre de C, dans la troisième case on trouvera le pourcentage correspondant à la proportion entière du nombre de G et dans la quatrième case on trouvera le pourcentage correspondant à la proportion entière du nombre de T. Par exemple,* composition("ATTGC") *renverra le tableau* {20,20,20,40}*. Pour rappel, pour calculer le pourcentage correspondant à la proportion entière du nombre de lettres, on prend le nombre de fois que la lettre apparaît, on le multiplie par 100 et on divise par le nombre total de lettres.*

5. *Écrire une procédure* caracteristiques *qui prend en argument une chaîne de caractères (que l'on supposera encodant une base azotée) et qui affiche sur la première ligne* Complement: *puis son complément et qui affiche ensuite sur la ligne suivante* Compositions : p% de A, q% de C, r% de G et s% de T *où p est le pourcentage correspondant à la proportion entière du nombre de A dans la base, q est le pourcentage correspondant à la proportion entière du nombre de C, r est le pourcentage correspondant à la proportion entière du nombre de G et s est le pourcentage correspondant à la proportion entière du nombre de T. Par exemple,* caracteristiques ("ATTGC") *affichera :*
   Complement: TAACG
   Compositions : 20% de A, 20% de C, 20% de G et 40% de T

□

Exercice 5 *On rappelle que pour un entier n strictement positif, ses diviseurs sont les nombres entiers entre 1 et n inclus qui le divisent. Un tel entier n est dit premier, si ses seuls diviseurs sont 1 et n (on rappelle que par convention 1 n'est pas premier). Pour toutes les fonctions demandées qui suivent, on supposera que les entiers donnés en arguments sont strictement positifs et on ne doit pas effectuer cette vérification.*

1. *Écrire une fonction* nbrDiviseurs *qui prend un entier naturel strictement positif comme argument et renvoie le nombre de ses diviseurs. Par exemple,* nbrDiviseurs (10) *renverra* 4 *car* 10 *a quatre diviseurs qui sont* $1, 2, 5$ *et* 10*.*

2. *Écrire une fonction* diviseurs *qui prend un entier naturel strictement positif comme argument et renvoie un tableau d'entiers contenant ces diviseurs. Par exemple,* diviseurs (10) *renverra le tableau* {1,2,5,10}*.*

3. *Écrire une fonction* premier *qui prend un entier naturel strictement positif comme argument et renvoie le booléen* true *si l'entier est premier et* false *sinon.Par exemple,* premier(10) *renverra* false *.*

4. *Écrire une procédure* listePremiers *qui prend en arguments un entier strictement positif et affiche tous les nombres premiers qui lui sont inférieurs ou égaux séparés par des espaces. Par exemple,* listePremiers (10) *affichera : 2 3 5 7.*

5. *Écrire une fonction* pgcd *qui prend en arguments deux entiers strictement positifs et renvoie leur pgcd, c'est-à-dire leur plus grand diviseur commun. Par exemple,* pgcd(8,12) *renverra* 4*.*

□