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

Université Paris-Diderot – Samedi 24 Octobre 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)`). 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. *Quelle est la valeur des variables* x *et* y *à la fin de l'exécution des instructions suivantes ?*

```
int x=7;
int y=3;
y=y*x;
if(25 <=y   || y <=50){
  x = 2*x;
} else{
   x = x*x;
}
```

2. *On considère la fonction* fonc *dont la définition est donnée ci-dessous. Que renvoie* fonc(5) *?*

```
public static int fonc(int x){
  int r=0;
  for(int i=2;i<x;i=i+1){
    r=r+i;
  }
  return r;
}
```

3. *Quel est le contenu du tableau* tab *à la fin des instructions suivantes ?*

```
int[] tab=new int[6];
for(int i=5;i>2;i=i-1){
  tab[i]=i;
  tab[5-i]=i;
}
```

□

Exercice 2 *On donne le code ci-dessous :*

```
public static (A) fonc1 ((B) a, (C) b){
  (D) d = "";
  if ((a >= b)) {
    d = "A";
  } else {
    d = "B";
  }
```

```
8     return d;
9   }
10
11  public static (E) fonc2 ((F) x, (G) y) {
12    (H) a = "HELLO";
13    (I) h=fonc1(a.length(),x);
14    h = h + h;
15    if (!stringEquals(h, y)){
16      System.out.println("Affichage 1");
17    } else {
18      System.out.println("Affichage 2");
19    }
20  }
```

1. *Donner les types manquants correspondant aux lettres (A),(B),(C),(D),(E),(F),(G),(H) et (I) dans le code parmi les types* int, String *et* void.
2. *Quelles valeurs peuvent être données aux arguments de la fonction* fonc2 *pour qu'elle affiche "Affichage 2".*

□

Exercice 3 *On dispose d'une fonction* g *dont la signature est* int g(int a, int b) *que l'on souhaite analyser.*

1. *Donner une suite d'instructions (qui seraient celles que l'on mettrait dans le programme principal* main*) permettant d'afficher les valeurs de* g *:*
   — *pour* a *qui vaut* 0 *et* b *qui vaut* 10*, puis,*
   — *pour* a *qui vaut* 20 *et* b *qui vaut* -15*, puis,*
   — *pour* a *qui vaut* 3 *et* b *qui vaut* 12.
2. *Donner une suite d'instructions permettant d'afficher les valeurs de* g *pour* a *qui vaut* 2 *et et* b *qui va de* 0 *à* 999.
3. *Donner une suite d'instructions qui affiche combien de fois la valeur de la fonction* g *est strictement négative pour* a *allant de* 5 *à* 205 *et* b *qui vaut* 1.
4. *Donner une suite d'instructions qui tire au hasard* 100 *entiers* n *compris entre* 0 *et* 1000 *et qui affiche pour chaque entier la valeur de* g(n,n) *(on ne devra pas vérifier que les entiers tirés au hasard sont tous différents). On pourra pour cela se servir d'une fonction* int randRange(int x, int y) *qui renvoie à chaque appel un entier* m *tiré au hasard tel que* x$\leq m <$ y.
5. *Donner une suite d'instructions qui affiche la plus petite valeur de la fonction* g *pour* a *allant de* -10 *à* 10 *et* b *allant de* 0 *à* 100.

□

Exercice 4 *Dans cet exercice, on s'intéresse à l'encodage en base 3 des entiers naturels sur une chaîne de caractères de taille 5. Une telle chaîne contient 5 entiers valant chacun 0, 1 ou 2. Ainsi une chaîne de caractères "*$a_0a_1a_2a_3a_4$*" composée uniquement de 0, 1 et 2 permet d'encoder l'entier :* $a_0 + a_1 * 3 + a_2 * 3^2 + a_3 * 3^3 + a_4 * 3^4$*. Par exemple, l'encodage de l'entier 5 est donné par la chaîne "21000". Avec cet encodage on peut donc coder les entiers de* 0 *à* 242.

1. *Écrire une fonction* isTernaryEncoding *qui prend comme argument une chaîne de caractères et teste si elle correspond au codage présenté, c'est à dire si sa taille est 5 et si elle ne contient que des 0, 1 ou des 2. Dans le cas positif elle renverra* true*, et sinon* false *. Par exemple,* isTernaryEncoding("10120") *renverra* true *et* isTernaryEncoding("10ba0") *renverra* false *et* isTernaryEncoding("102") *renverra* false *.*
2. *Écrire une fonction* powerThree *qui prend comme argument un entier* n *(supposé positif) et renvoie* 3^n*.*
3. *Écrire une fonction* decode *qui prend comme argument une chaîne de caractères, et si cette chaîne correspond à notre encodage renvoie la valeur entière correspondante et sinon elle renvoie -1. Par exemple,* decode("12000") *renverra* 7, decode("10001") *renverra* 82 *et* decode("1201") *renverra* -1.
4. *Écrire une fonction* encode *qui prend comme argument un entier et renvoie une chaîne de caractères de taille 5 correspondant à son encodage en base 3. Si l'entier est négatif ou supérieur ou égal à 243, cette fonction renvoie la chaîne vide "". Par exemple* encode(7) *renvoie* "12000" *et* encode(12) *renvoie* "01100". Indice : *Si l'on a un entier* n *et on veut connaître son encodage "*$a_0a_1a_2a_3a_4$*", on a* a_0 *qui vaut* n % 3, a_1 *qui vaut* (n / 3) % 3, a_2 *qui vaut* ((n / 3))/3 % 3 *etc.*

□

Exercice 5 *Dans cet exercice, on veut créer des fonctions pour générer automatiquement des chaînes de caractères.*

1. *Écrire une fonction* repeat *qui prend comme arguments une chaîne de caractères* st *et un entier* n *supposé positif et renvoie la chaîne de caractères correspondant à* n *répétitions consécutives de* st*. Par exemple* repeat("bla",3) *renverra la chaîne "*blablabla*".*

2. *Écrire une fonction* mix *qui prend comme arguments deux chaînes de caractères* st1 *et* st2 *et qui renvoie la chaîne de caractères avec* st2 *suivi de* st1 *si* st1 *et* st2 *ne font pas la même longueur et sinon la chaîne où l'on alterne consécutivement une lettre de* st1 *puis une lettre de* st2*. Par exemple* mix("bla","bouh") *renverra la chaîne* "bouhbla" *et* mix("bla","woh") *renverra la chaîne* "bwloah".
3. *Écrire une fonction* push *qui prend comme argument une chaîne de caractères* st *et renvoie la chaîne de caractères de même longueur de* st *où la dernière lettre de* st *devient la première lettre, la première lettre de* st *devient la deuxième etc et l'avant dernière lettre de* st *devient la dernière lettre. Si* st *vaut* "" *alors la fonction renvoie* ""*. Par exemple* push(" partiel ") *renverra la chaîne* " lpartie ".
4. *Écrire une fonction* multipush *qui prend comme arguments une chaîne de caractères* st *et un entier* n *supposé positif et renvoie la chaîne de caractères de même longueur de* st *correspondant à* n *appels de* push *sur la chaîne* st*. Par exemple,* multipush(" partiel ",2) *renverra la chaîne* " elparti ".

□

Exercice 6 *Le Bingo est un jeu de hasard à plusieurs joueurs. Il se joue avec 90 boules numérotées. Chaque joueur possède un carton comportant 15 numéros compris en 1 et 90. Le but du jeu est d'être le premier à marquer les numéros contenus sur le carton conformément aux boules qui sont tirées par le maître du jeu. On représente ici les cartons au moyen de tableaux d'entiers de longueur 15. Ce tableau contient les numéros qui n'ont pas encore été tirés (compris entre 1 et 90) et des 0 (qui remplacent les numéros qui ont été tirés). Ainsi quand un tableau n'a pas encore eu de numéro tiré, il ne contient pas de 0. Dans notre version, un numéro peut apparaître plusieurs fois dans un tableau. On souhaite dans cet exercice écrire des fonctions intermédiaires permettant de programmer un tel jeu.*

1. *Écrire une fonction* isCard *qui prend en argument un tableau d'entiers et vérifie qu'il s'agit bien d'un carton de Bingo vierge (i.e. qui contient 15 entiers (pas forcément tous différents) compris entre 1 et 90 et aucun 0) et renvoie* true *si c'est bien le cas,* false *sinon. Par exemple, si l'on a* int [] card1={18,15,1,4,15}*,* int [] card2 ={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} *et* int [] card3 ={140,2,3,4,5,6,7,8,9,10,11,12,13,14,15} *, alors* isCard(card1) *et* isCard(card3) *renverront* false *et* isCard(card2) *renverra* true.
2. *Écrire une fonction* mark *qui prend en arguments un carton de Bingo* card *(dont certains numéros ont possiblement déjà été marqués) et un entier* n *correspondant au tirage d'une boule et renvoie une copie de* card *dans laquelle toutes les occurrences de* n *on été marquées, c'est à dire remplacées par* 0*. Par exemple, si l'on a* int [] card={10,2, 3,4,5,6,7,8,9,10,11,12,13,14,10} *, alors* mark(card,10) *renverra* {0,2,3,4,5,6,7,8,9,0,11,12,13,14,0} .
3. *Le Texas Bingo est une variante du jeu dans laquelle le premier coup est joué différemment : on marque tous les numéros ayant la même parité (pair ou impair) que le numéro tiré. Écrire une fonction* texasMark *qui prend en arguments un carton de Bingo* card *et un entier* n *correspondant au tirage d'une première boule et renvoie une copie de* card *dans laquelle tous les numéros ayant la même parité que* n *on été remplacés par* 0*. Par exemple, si l'on a* int [] card ={10,2,3,4,5,6,7,8,9,10,11,12,13,14,10} *, alors* texasMark(card,42) *renverra* {0,0,3,0,5,0,7,0,9,0,11,0,13,0,0} .
4. *Le Numéro Sauvage est une variante du jeu dans laquelle le premier coup est joué différemment : on marque tous les numéros ayant le même chiffre d'unités que le numéro tiré (le chiffre d'unités étant le dernier chiffre d'un nombre lorsqu'on le lit de gauche à droite). Écrire une fonction* wildMark *qui prend en argument un carton de Bingo* card *et un entier* n *correspondant au tirage d'une première boule et renvoie une copie de* card *dans laquelle tous les numéros ayant le même chiffre d'unités que* n *on été remplacés par* 0 *. Par exemple, si l'on a* int [] card ={32,52,3,4,5,6,7,8,9,42,11,12,13,14,10} *, alors* wildMark(card,12) *renverra* {0,0,3,4,5,6,7,8,9,0,11,0,13,14,10} .
5. *Écrire une fonction* win *qui prend en argument un carton de Bingo* card *et vérifie que toutes ses cases sont marquées, renvoyant* true *si c'est bien le cas,* false *sinon. Par exemple, si l'on a* int [] card1 ={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0} *et* int [] card2 ={0,0,3,0,5,0,7,0,9,0,11,0,13,0,0} *alors* win(card1) *renverra* true *et* win(card2) *renverra* false .
6. *On considère maintenant un objectif intermédiaire consistant à marquer une suite consécutive de nombres de longueur* m*. Écrire une fonction* smallwin *qui prend en argument un carton de Bingo* card *et un entier* m *et vérifie que* card *contient une séquence de cases marquées de longueur* m*, renvoyant* true *si c'est bien le cas,* false *sinon. Par exemple, si l'on a* int [] card1 ={0,0,3,0,5,0,7,0,9,0,11,0,13,0,0} *et* int [] card2 ={0,0,3,0,5,0,7,0,0,0,11,0,13,0,0} *alors* smallwin(card1,3) *renverra* false *et* smallwin(card2,3) *renverra* true *(car dans* card2 *on a bien 3 zéros consécutifs mais pas dans* card1*).*

□