II.1102 – Algorithmique et Programmation
Examen écrit

Patrick Wang

20 Janvier 2020

1 Consignes générales

— La durée de cet examen est de **3 heures**.
— Prenez le temps de lire l'énoncé dans son intégralité et assurez vous que le sujet comporte bien **5 pages**.
— Le sujet est à rendre avec la copie.
— Aucun document n'est autorisé pour cet examen.
— Ce partiel est noté sur 30.
— Attention à ne pas confondre **afficher** et **retourner**.
— Veuillez faire attention à la syntaxe utilisée en Java. Quelques erreurs pourront être tolérées mais un oubli systématique des règles de syntaxe sera pénalisé.

2 Questions de compréhension (10 points)

2.1 Questions de cours (5 points)

1. Quelles sont les différences entre les types primitifs `short`, `int`, et `long` ?
2. Quelle est la différence entre une boucle `while` et une boucle `do while` ?
3. Qu'est-ce qu'une classe ? Qu'est-ce qu'un objet ?
4. Quels sont les quatre types de visibilité ? Quelles sont les caractéristiques de chacun de ces types ?
5. Que signifie le mot clé `static` lorsqu'il est mis devant un attribut ? Que signifie le mot clé `final` lorsqu'il est mis devant un attribut ? Que signifie la combinaison `static final` lorsqu'elle est mise devant un attribut ?

2.2 Compréhension de code (5 points)

2.2.1 Question 1

Lisez attentivement le programme ci-après :

```
1 public class Main {
2     public static void main(String[] args) {
3         int i = 2020;
4         System.out.println(i);
5         premiereFonction(i);
```

```
        System.out.println(i);

        // Suite de la methode main() apres saut de page
        int[] tableau = {1, 3, 5};
        System.out.println(tableau[1]);
        secondeFonction(tableau);;
        System.out.println(tableau[1]);
    }

    public static void premiereFonction(int n) {
        if (n % 2 == 0) {
            n = 3 * n;
        } else {
            n = 2 * n;
        }
        System.out.println(n);
    }

    public static void secondeFonction(int[] array) {
        if (array.length > 2) {
            array[1] = 0;
            System.out.println(array[1]);
        } else {
            System.out.println("Erreur");
        }
    }
}
```

Question :

1. Que fait ce programme ? Indiquez, dans l'ordre, les différents messages affichés en console.

2.2.2 Question 2

```
public static int uneAutreFonction(int n) {
    int a = 0;
    int b = 1;
    int c = 0;
    int i = 0;
    while (i <= n) {
        if (i < 2) c = i;
        else {
            c = a + b;
            a = b;
            b = c;
            i++;
        }
    }
    return c;
}
```

Question :

1. Que fait la fonction `uneAutreFonction()` ?

3 Algorithmique (15 points)

3.1 Algorithme de recherche dichotomique (4 points)

La recherche dichotomique est un algorithme de recherche d'un élément dans un **tableau trié**. Son principe est le suivant :

1. On coupe le tableau trié en son milieu ;
2. On compare la valeur recherchée avec l'élément médian :
   — Si la valeur recherchée est strictement supérieure à l'élément médian, on va prendre la moitié de tableau à droite de cet élément médian ;
   — Si la valeur recherchée est strictement inférieure à l'élément médian, on va prendre la moitié de tableau à gauche de cet élément médian ;
   — Si la valeur recherchée est égale à l'élément médian, l'algorithme se termine.
3. On continue tant que l'élément n'est pas trouvé ou que la partie retenue du tableau est vide.

Les instructions suivantes sont toutes utilisées pour implémenter l'algorithme de recherche dichotomique.

```
1  int indiceDroit = tableau.length - 1;
2  } else {
3  return null;
4  } else if (tableau[indiceMilieu] > valeurRecherchee) {
5  indiceDroit = indiceMilieu - 1;
6  while (indiceGauche <= indice Droit) {
7  }
8  }
9  if (tableau[indiceMilieu] < valeurRecherchee) {
10 indiceGauche = indiceMilieu + 1;
11 int indiceMilieu = (indiceGauche + indiceDroit) / 2;
12 return indiceMilieu;
13 int indiceGauche = 0;
```

Questions :

1. Remettez les instructions dans le bon ordre. Au lieu de simplement utiliser les numéros de ligne, veillez à recopier entièrement les instructions.

3.2 Coefficients binomiaux ou Triangle de Pascal (4 points)

Pour rappel, le triangle de Pascal se présente sous la forme suivante :

```
   0   1   2   3   4   5   6
 ----------------------------
0|  1
1|  1   1
2|  1   2   1
3|  1   3   3   1
4|  1   4   6   4   1
5|  1   5   10  10  5   1
6|  1   6   15  20  15  6   1
```

Chaque élément du triangle de Pascal représente un coefficient binomial. Par exemple, l'élément situé en ligne 3 et en colonne 2 est égal à $\binom{3}{2}$, aussi noté C_3^2.

Les coefficients binomiaux peuvent être calculés grâce à la relation de récurrence suivante :

$$\forall(n,\, k) \in \mathbb{N}, \quad 0 < k < n, \quad \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Lorsque la condition sur k et n n'est pas vérifiée, alors le coefficient binomial est égal à 1.

Questions :

1. Écrivez une fonction récursive permettant de calculer et de **retourner** la valeur de $\binom{n}{k}$.

3.3 Algorithme de tri par insertion (7 points)

Le tri par insertion est l'algorithme que la plupart des personnes vont utiliser naturellement pour trier un paquet de cartes. Son principe est le suivant :

Dans l'algorithme de tri par insertion, on parcourt le tableau à trier du début à la fin. Au moment où l'on considère le i-ème élément, les éléments qui le précèdent sont déjà triés.

L'objectif d'une étape est d'insérer le i-ème élément à sa place parmi ceux qui le précèdent. Il faut pour cela trouver où l'élément doit être inséré en le comparant aux autres, puis décaler les éléments afin de pouvoir effectuer l'insertion.

En pratique, ces deux actions sont fréquemment effectuées en une passe, qui consiste à faire « remonter » l'élément au fur et à mesure jusqu'à rencontrer un élément plus petit que l'élément à trier.

Questions :

1. Écrivez l'algorithme de tri par insertion.
2. Rappelez la définition de la complexité algorithmique en temps.
3. Pour le tri par insertion, quelle configuration du tableau amène au pire cas ? Quelle est la complexité dans le pire cas ?
4. Pour le tri par insertion, quelle configuration du tableau amène au meilleur cas ? Quelle est la complexité dans le meilleur cas ?

4 Modélisation (5 points)

Dans cet exercice, on souhaite modéliser une version simplifiée de Moodle. Moodle est accessible par les enseignants et les étudiants de l'ISEP. Ces utilisateurs sont identifiés par leurs adresses email, en plus d'avoir les informations classiques tels que le nom et le prénom.

Sur Moodle, on peut trouver un catalogue de cours. Les cours appartiennent à l'un des quatre domaines d'enseignement suivant : Informatique, Télécom, Électronique, Sciences des données. Un cours est constitué d'une liste de fichiers de cours et d'un forum comportant des messages écrits soit par un étudiant soit par un enseignant. Ces messages ont aussi une date de publication et un contenu.

Un utilisateur de Moodle (enseignant ou étudiant) peut s'inscrire à un ou plusieurs cours et consulter les fichiers présents. Cependant, seuls les étudiants ont des notes pour chaque cours.

Questions :

1. Représentez une classe et une classe abstraite dans un diagramme UML de classe. Rappelez la différence entre une classe et une classe abstraite.
2. Rappelez les symboles utilisés pour représenter les différents types de visibilité utilisés en programmation orientée objet.

3. Rappelez les types de flèches utilisées pour représenter l'héritage, l'implémentation, l'agrégation, et la composition.
4. Représentez le diagramme UML de classe permettant de modéliser cette version simplifiée de Moodle.