Examen

2ème année CPI

UEF4.3. Programmation Orientée Objet

Durée 02 heures

Documents interdits

Exercice 1 (5 points)

Indiquer l'affichage du programme suivant :

```
class Exercice1 {

public static void
main(String[] args) {
  D d1 = new D();
  D d2 = new D(true);
  E e1 = new E();
  E e2 = new E(false);
  C c = new C();

 d1.m1();
 d2.m1();
 d2.m3();
 e1.m1();
 e1.m3();
 e2.m3();
 c.m2();

System.out.println(d1.n);
System.out.println(d1.b);
System.out.println(d1.i);
System.out.println(d2.n);
System.out.println(d2.b);
System.out.println(d2.i);
System.out.println(e1.n);
System.out.println(e1.b);
System.out.println(e1.o);
System.out.println(e2.n);
System.out.println(e2.b);
System.out.println(e2.o);
System.out.println(c.n);
System.out.println(c.d);

    }
}
```

```
abstract class A {
    static int n = 0;
    A() {  n++;    }
    A(boolean b) {
        if (b)   n++;
      }
}

abstract class B extends A {
    boolean b = false;
    B() { super();  }
    B(boolean b ) {
        super(b);
        n++;
    }
    abstract void m1();
}

class C extends A {
    int d = 0;
    C() { super(true); }
    void m1() {
      if (d == 1)
            d++;
             }
    void m2() {
        if (d == 2)
            d--;
       }
}

class D extends B {
    int i = 1;
    D() {
        super();
        b = false;
    }
```

```
D(boolean b) {
        super(b);
        n++;
        i++;
    }
    void m1() {

        if (b)   b = false;
        else     b = true;

    }
    void m3() {
        i++;
    }
}
class E extends B {
    int o;
    E() {
        super();
        o++;
    }
    E(boolean b) {
        super(b);
        o = 4;
    }
    void m1() {
     o--;

    }
    void m3() {
        o++;
    }
}
```

Exercice 2 (8 points).

Un brin d'ADN est une séquence formée de 4 bases azotées : L'adénine (A), la cytosine (C), la guanine (G) et la thymine (T).

Exemple : **GCTATTCGAG**

Un double-brin d'ADN est constitué de 2 brins d'ADN de même longueur, parallèles et liés. L'adénine se lie toujours avec la thymine et la cytosine avec la guanine.

Exemple :

Ecrire une classe BrinADN comportant :

- Un attribut appelé brinAdn de type chaine de caractères.
- Un constructeur ayant comme paramètre une chaine de caractères et qui construit un objet de type BrinADN si cette chaine est une séquence ADN valide.
- Une méthode `public boolean sequenceValide(string S)` qui vérifie si une séquence donnée est valide ou pas. Dans le cas où la chaine passée au constructeur n'est pas conforme (contient un autre caractère que A, C, G et T), le constructeur lance une exception de type `SequenceException` qui doit être définie.
- Une méthode `public void sousSequence (String s)` qui vérifie si une séquence *s* donnée est une sous-séquence de la séquence brinAdn et affiche le résultat. Pour résoudre cette question utiliser la méthode `String substring (int beginIndex, int endIndex)` qui revoie la sous-chaine de caractères comprise entre beginIndex et endIndex.
- Une méthode `public void doubleBrinAdn (String s)` qui vérifie si une séquence donnée peut former un double brin d'ADN avec la séquence brinAdn et affiche le résultat.
- Une méthode `public double pourcentageBase (char car)` qui reçoit en paramètre un caractère représentant une des bases azotées (A, C, G ou T) et retourne le pourcentage de présence de la base dans la séquence brinADN. Cette méthode lance une exception de type BaseException dans le cas où le caractère passé en paramètre n'est pas conforme. L'exception BaseException doit être définie.
  Exemple : dans la séquence **GCTATTCGAG** il y a 20% de cytosine

Pour tester la classe BrinADN, la classe main doit recevoir en arguments de la ligne de commande deux chaines de caractère s_1 et s_2 et un caractère *car* et effectuer les opérations suivantes :

1. Construire un objet *brin* de type BrinADN avec la séquence s_1.

2. Vérifier si la séquence s_2 est une sous-séquence de l'attribut brinAdn de l'objet *brin* construit précédemment.

3. Vérifier si la séquence s_2 peut former un double brin avec l'attribut brinAdn de l'objet *brin*.

4. Calculer le pourcentage de présence de la base azotée correspondant au caractère *car* donné en argument.

Exercice 3 (7 points)

On désire modéliser une pile et une file d'entiers en créant deux classes Pile et File implémentant toutes les deux l'interface ListeChainee définie comme suit :

```
interface ListeChainee {

Void insérer (Element e) ;

Void supprimer () ;

boolean estVide() ;

}
```

Les éléments de la pile et de la file sont de type Element défini comme suit :

```
class Element {

int valeur ;

Element suivant ;

// constructeur à écrire

}
```

1. Ecrire une classe Pile ayant comme attribut *sommet* de type Element. Implémenter les méthodes nécessaires.

2. Ecrire une classe File ayant comme attribut *tete* de type Element. Implémenter les méthodes nécessaires.

3. Dans la méthode main effectuer les opérations suivantes :

a- Créer un objet de type Pile et un objet de type File.

b- Ouvrir un fichier de type texte contenant des entiers séparés par des blancs. Copier son contenu dans un tableau *tab* d'entiers de taille 100.

c- Insérer les éléments du tableau *tab* dans la file créée précédemment.

d- Trier les éléments du tableau et les insérer dans la pile de façon à ce que le nombre le plus petit soit en sommet de pile.

NB : La gestion des exceptions pouvant être déclenchées par les opérations d'entrées sorties doit être prise en compte.