Université Henri Poincaré (Nancy1)
DEUG Science 2 – MIAS
Epreuve d'informatique
Session de janvier 2005
Date : 05 janvier 2005
Horaire : de 14h00 à 16h00

Faculté des Sciences
Sujet d'examen
Durée du sujet : 2 heures
Responsable : Laurent Noé
laurent.noe@loria.fr

Calculatrices autorisées.
Documents autorisés.

Les exercices sont indépendants. Les temps donnés sont indicatifs.

1–Questions de cours (15 minutes)

Répondez brièvement aux questions suivantes :

1. Donnez le code java d'une classe qui possède au moins une variable d'instance, au moins un constructeur, et au moins une méthode autre que le constructeur (prenez par exemple la classe `Point`).
2. Que signifie le terme *" durée de vie d'une variable"* ? Expliquez le cas des *variables locales* aux méthodes, et des *variables d'instances* des objets.
3. Que savez vous sur la classe `Object` ?

2–QCM (30 minutes)

n'utilisez que la feuille réponse

Vous devez effectuer un ou plusieurs choix parmi ceux proposés. Vous préciserez éventuellement les raisons de vos choix.

Question 1 Quelle relation entre la méthode `p1.equals(p2)` et l'instruction `p1 == p2` ?

a. `p1 == p2` $\Longrightarrow$ `p1.equals(p2)`
b. `p1.equals(p2)` $\Longleftrightarrow$ `p1 == p2`
c. `p1.equals(p2)` $\Longrightarrow$ `p1 == p2`
d. elles sont indépendantes

Question 2 Quelle instruction possible pour compléter la méthode factorielle ?

```
int fact(int n) {
  int r = 1;
  for( int i = 1; i <= n ; i++) {
     /* A completer */
  }
  return r;
}
```

a. `r = r + i;`
b. `r = r * i/(i-1);`
c. `r = r * i;`
d. `r = i * r/(r-1);`

Question 3 A quoi sert le mot clef `return` dans l'instruction `return a;` ?

a. Il arrête l'exécution de la méthode en cours
b. Il renvoie la valeur actuelle de la variable `a`
c. Il renvoie le nom actuel de la variable `a`
d. Il appelle la méthode `a()`

Question 4 On code les entiers naturels en utilisant un codage sur 8 bits avec complement à deux. Quel est alors le codage hexadécimal de $+23_d$

a. 35_h
b. EA_h
c. 17_h
d. 42_h

Question 5 On utilise un code en virgule fixe sur 16 bits (8 bits / 8bits). Quelle est la valeur codée par `1A33`

a. environ `26,50`
b. environ `26,20`
c. environ `27,20`
d. environ `27,50`

Question 6 Comment réaliser une multiplication d'un entier positif x par 320_d sans utiliser l'opérateur de multiplication ? (rappel l'opérateur de décalage des bits à gauche est donné par "<<". Exemple : x = x << 3 est équivalent à x = x * 8)

a. `x = x << 8 + x << 6`

b. `x = x << 320`

c. `x = x << 7 + x << 5`

d. `x = x << x`

Question 7 Que signifie le mot clef `static` devant une méthode ?

a. La méthode est privée

b. La méthode a un codomaine (type du résultat) vide

c. Il n'est pas nécessaire de construire un objet de la classe pour appeler cette méthode

d. Il est obligatoire de construire un objet de la classe pour appeler cette méthode

Question 8 Quelles sont les fonctions récursives terminales ?

a.

```
int f1(int n){
  if (n == 0)
     return 1;
  else
     return f1(n-1) * n;
}
```

b.

```
int f2(int n, int a){
  if (n == 0)
    return a;
  else
    return f2(n-1,a*n);
}
```

c.

```
double g1(double x,int n,double a,double b){
  if (n == 0)
     return a;
  if (n%2 == 0)
    return g1(x, n/2, a, b*b);
  else
    return g1(x, n/2, a*b, b*b);
}
```

d.

```
double g2(double x, int n){
  double r = 1, b = x;
  while(n > 0){
    if (n%2 != 0)
       r*=b;
    n/=2;
    b*=b;
  }
  return r;
}
```

3–Dominos (40 minutes)

On vous propose de définir une classe qui modélise une chaîne de dominos. Un `Domino` est représenté par deux entiers `valGauche` et `valDroite`. Une chaîne de dominos `ChaineDeDominos` est représentée par une liste d'objets de type `Domino`. Cette chaîne n'est valide que si, pour chaque domino de la liste, la valeur gauche du dit domino est égale à la valeur droite du domino précédent dans la liste.

```
public class Domino {
  public int valGauche;
  public int valDroite;

  public Domino(int valGauche, int valDroite){
    this.valGauche = valGauche;
    this.valDroite = valDroite;
  }

  public boolean peutSuivre(Domino  precedent) {
    return this.valGauche == precedent.valDroite;
  }

  public boolean peutPreceder(Domino  suivant) {
    return           /* a completer */          ;
  }

  public String toString(){
    return " [ " + valGauche +
           " , " + valDroite + " ] ";
  }

  /*
   * A completer
   */
}
```

```
public class ChaineDeDominos {
  public LinkedList chaineDeDominos = null;

  public ChaineDeDominos () {
    chaineDeDominos = new  LinkedList();
  }

  /*
   * A completer
   */

}
```

La documentation des méthodes des classes suivantes vous est donnée :

classe `LinkedList`
- `void add(Object el)` : ajoute un élément à la fin de la liste.
- `void remove(int i)` : retire le $i + 1^{eme}$ élément de la liste.
- `Object get(int i)` : donne le $i + 1^{eme}$ élément de la liste.
- `Object getLast()` : donne le dernier élément de la liste.
- `int size()` : donne le nombre d'éléments de la liste.
- `Iterator Iterator()` : donne un itérateur sur la liste qui débute au premier élément.

classe `ListIterator`
- `boolean hasNext()` : retourne `true` si l'élément courant possède un suivant, `false` sinon.
- `Object next()` : donne l'élément suivant dans la liste.
- `void remove()` : supprime le dernier élément retourné par la méthode `next()`. Elle ne produit pas d'effets de bord sur l'itérateur en cours.

Les éléments d'une liste sont indexés à partir de 0.
L'opérateur de concaténation de chaînes de caractères est l'opérateur +
Faites **attention** aux coercions de type (cast) lorsque vous utilisez les méthodes `next()`, `get(int i)` et `getLast()`.

Question 1 Dans la classe `Domino`, complétez la méthode `boolean peutPreceder(Domino suivant)` qui indique si la valeur droite du domino courant `this` est égale à la valeur gauche du domino `suivant`.

Question 2 Dans la classe `Domino`, définissez une méthode `void tourne()` qui permute les valeurs gauche et droite du domino courant.

Question 3 Dans la classe `Domino`, définissez une méthode `boolean equals(Domino autre)`. Deux dominos "`[a,b]`" et "`[b,a]`", bien qu'orientés de manière différente, seront supposés égaux.

Question 4 Dans la classe `ChaineDeDominos`, définissez une méthode `String toString()` qui retourne une chaîne de caractères à partir de la chaîne de dominos (comme par exemple "`[1,5] [5,2] [2,1]`"). Vous utiliserez un itérateur, ainsi que la méthode `String toString()` de la classe `Domino`.

Question 5 Comme un jeu de domino ne possède pas deux fois la même pièce, il faut vérifier si un domino "`[a,b]`" est déja présent dans la chaîne (sous sa forme "`[a,b]`" ou "`[b,a]`"). Dans la classe `ChaineDeDominos`, définissez une méthode `boolean dejaPresent(Domino d)`. Vous utiliserez un itérateur, ainsi que la méthode `boolean equals(Domino autre)` de la classe `Domino`.

Question 6 Dans la classe `ChaineDeDominos`, définissez une méthode `boolean ajoute(Domino d)` qui ajoute un domino `d` en fin de chaîne selon les règles suivantes :
- le domino n'est pas présent dans la chaîne (utiliser la méthode `boolean dejaPresent(Domino d)`).
- le domino `d`, s'il est ajouté, ne remet pas en cause la validité de la chaîne.
- le domino `d` peut éventuellement être tourné avant d'être ajouté.
- la méthode renvoie (retourne) la valeur `true` si l'ajout de `d` est réussi, `false` sinon.

Question 7 Dans la classe `ChaineDeDominos`, définissez une méthode `boolean enleve(int i)` qui enlève le domino à la position i et, si nécessaire, la plus petite sous-chaîne de ses successeurs. Il faut obligatoirement que la chaîne reste valide après cette modification. Vous utiliserez un itérateur. L'indice i devra être tel que $0 \leq i < chaineDeDominos.size()$ sinon la fonction retournera la valeur `false`.

Exemple : si `chaîne = [3,2][2,5][5,0][0,0][0,2][2,6]` alors
`chaine.enleve(3)` donnera `chaîne = [3,2][2,5][5,0][0,2][2,6]` et
`chaine.enleve(1)` donnera `chaîne = [3,2][2,6]`.

4–Héritage (30 minutes)

n'utilisez que la feuille réponse

Question 1 à 6 Indiquez ce qu'affiche l'exécution de la commande `java App`. Vous pouvez éventuellement compléter les schémas afin de vous aider

```
public class A {

  public int cH ;
  public int cR ;

  public A(){
    cH = 1;
    cR = 2;
    System.out.println("Constructeur de A");
    System.out.println(" -cH = "+cH);
    System.out.println(" -cR = "+cR);
  }

  public void mH(){
    System.out.println("mH() de A");
    System.out.println(" -cH = "+cH);
    System.out.println(" -cR = "+cR);
  }

  public void mR(){
    System.out.println("mR() de A ");
    System.out.println(" -cH = "+cH);
    System.out.println(" -cR = "+cR);
    }
}
```

```
public class B extends A {

  public int cR;

  public B(){
    cH = 3;
    cR = 4;
    System.out.println("Constructeur de B");
    System.out.println(" -cH= "      +cH);
    System.out.println(" -cR= "      +cR);
    System.out.println(" -super.cR= "+super.cR);
  }

  public void mR(){
    cH = 0;
    if (super.cH == 0)
      super.mR();
    System.out.println("mR() de B");
    System.out.println(" -cH= "      +cH);
    System.out.println(" -cR= "      +cR);
    System.out.println(" -super.cR= "+super.cR);
  }
}
```

```
public class App {

    public static void main(String[] args){

      System.out.println("Question 1");
        A a = new A();

      System.out.println("Question 2");
        B b = new B();

      System.out.println("Question 3");
        a.mR();

      System.out.println("Question 4");
        a.cR = 5;
        System.out.println("     b.cR = "+      b.cR);
        System.out.println("((A)b).cR = "+ ((A)b).cR);

      System.out.println("Question 5");
        b.mH();

      System.out.println("Question 6");
        ((A)b).mR();
    }
}
```