Université Henri Poincaré (Nancy1)
DEUG Science 2 – MIAS
Epreuve d'informatique
Session de janvier 2003
Date : 14 janvier 2003
Horaire : de 13h30h à 15h30

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

Calculatrices non autorisées.
Documents limités à deux pages recto format A4.

Les exercices sont indépendants. Les temps donnés sont indicatifs.

1–Question de cours (15 minutes)

Répondez brièvement aux questions suivantes :

1. Donnez deux spécificités qui distinguent les *classes abstraites* des *interfaces* en java. Quelle est l'utilité des classes abstraites.
2. Qu'est ce que la *Généricité* ? Comment est-elle *simulée* en java ?
3. Question subsidiaire : quels éléments différencient une architecture *RISC* d'une architecture *CISC*.

2–Carnet d'adresses (30 minutes)

On vous propose de définir un classe qui modélise un carnet d'adresses. Un carnet d'adresses sera représenté par une liste d'objets de type `Lien`.

```
public class Lien{
  public String nom;
  public String adresse;

  public Lien(String nom, String adresse){
    this.nom     = nom;
    this.adresse = adresse;
  }
}
```

```
public class Carnet{
  public LinkedList connaissances = null;

  public Carnet(){ // crée un carnet vide
     connaissances = new  LinkedList();
  }

  /*
   * A completer
   */
}
```

La documentation vous donne par ailleurs des méthodes des classes suivantes :

classe `LinkedList`
- `void add(Object el)` : ajoute un élément à la fin de la liste.
- `int size()` : donne le nombre d'éléments dans la liste.
- `void remove(int i)` : retire le i^{eme} élément de la liste (le premier élément est à l'index 0).
- `Object get(int i)` : donne le i^{eme} élément de la liste (le premier élément est à l'index 0).
- `ListIterator listIterator()` : 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

classe `String`
- `boolean equals(String str)` : retourne true si la chaîne de caractères est la même que `str`.

Question 1 Définissez une méthode de profil `int nbLiens()` qui retourne le nombre de liens contenus dans le carnet d'adresses.

Question 2 Définissez une méthode de profil `void ajoute(Lien l)` qui ajoute un lien dans le carnet d'adresses.

Question 3 Définissez une méthode de profil `int nbAdresses(String nom)` qui retourne le nombre d'adresses mémorisées pour le `nom` donné (vous utiliserez un itérateur).

Question 4 Définissez une méthode de profil `String[] adresses(String nom)` qui retourne un tableau des différentes adresses pour le `nom` donné (vous utiliserez un itérateur).

3–QCM (35 minutes)

n'utilisez que la feuille réponse

Vous devez effectuer un ou plusieurs choix parmi ceux proposés. Vous préciserez éventuellement, selon la question, les raisons de vos choix (calcul, raisonnement suivi, aléas utilisé...)

Question 1 Quels types sont primitifs ?

a. `boolean` **b.** `int[]`
c. `char` **d.** `String`

Question 2 En utilisant un mot de 8 bits, que vaut le codage *hexadécimal* de la valeur décimale `-21` ?

a. `FFEA` **b.** `E9`
c. `EB` **d.** `EA`

Rappel : *le complémentaire d'un nombre binaire se calcule en inversant les bits* ***puis*** *en ajoutant 1*

Question 3 Donnez la ou les instructions suivantes valides :

```
LinkedList list = new LinkedList();
int a = 5;
/* A completer */
```

a. `list.add(a);` **b.** `list.add(new Integer(a));`
c. `list.add((Object)a);` **d.** `list.add((Object)new Integer(a));`

Question 4 On suppose avoir un objet instancié `list` de type `LinkedList` . Cette liste contient un certain nombre d'éléments de type `String`. Choisissez la ou les instructions suivantes valides :

a. `String s = list.get(0);` **b.** `String s = (LinkedList) list.get(0);`
c. `String s = ((String)list).get(0);` **d.** `String s = (String)list.get(0);`

Question 5 On code le nombre `10,10d` en virgule fixe sur deux octets (la partie "entière" est codée sur le premier octet, la partie "décimale" sur le second). Que vaut son codage *hexadécimal* ?

a. `0A0C` **b.** `0A33`
c. `0A19` **d.** `0A00`

Question 6 Pour quelles familles d'entiers cette fonction renvoie t-elle toujours `true` ?

```
static boolean f(int n) {
   while((n != 1)&&(n % 2 == 0)){
         n = n / 2;
   }
   return (n == 1);
}
```

a. les entiers n divisibles par 2.
b. les entiers strictement positifs n divisibles par 2.
c. les entiers n dont la valeur absolue est une puissance de 2.
d. les entiers strictement positifs n qui sont des puissances de 2.

Question 7 Quelles sont les fonctions récursives terminales ?

a.

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

b.

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

c.

```
static boolean g1(int n){
  if (n == 1)
     return true;
  if (n%2 == 0)
    return g1(n/2);
  else
    return g1(3*n+1);
}
```

d.

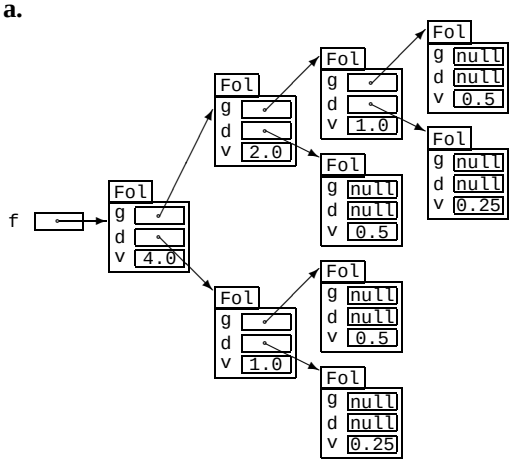
```
static boolean g2(int n){
  while( n != 1){
     if (n%2 == 0)
        n=n/2;
     else
        n=3*n+1;
  }
  return true;
}
```

Question 8 Quel est l'état mémoire juste après l'exécution de l'instruction contenue dans la fonction `main()`.

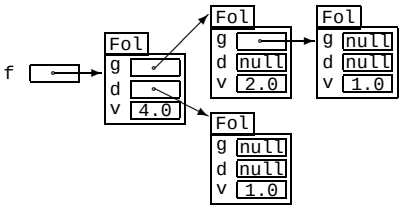
```
public class Fol{
    public Fol g = null;
    public Fol d = null;
    public double v;

    public Fol(double a){
        this.v = a;
        if (a >= 1){
            g = new Fol(a/2);
            d = new Fol(a/4);
        }
    }
}
```

```
public Class App {
    public static void main(String args[]){
        Fol f =new Fol(4.0);
    }
}
```

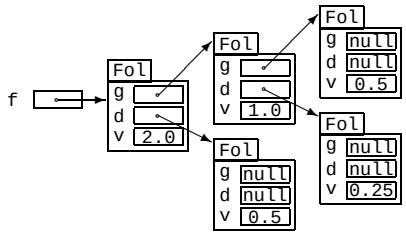
a.



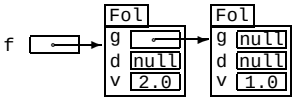
b.



c.



d.



4–Héritage (40 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 = 10;
    cR = 20;
    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 = 30;
    cR = 40;
    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();
   }
}
```