N1360

ISO JTC1/SC22/GT5 - FORTRAN

Cadarache, FRANCE 14 - 18 JUIN 1999

FORTRAN ORIENTÉ OBJETS
OBJECT - ORIENTED FORTRAN

Nécessité d'un paramètre de type dans ALLOCATE
Need of a type-like parameter in ALLOCATE

Syntaxe proposée / *proposed syntax*:

ALLOCATE (pointer, CAST = typmold)

AFNOR

N1360

Liste des classes Java
List of Java classes

`FLISTE`	listes lineaires d'objets / *linear lists of Objects* (avec ALLOCATE ordinaire / *with ordinary ALLOCATE*)
`FTestListe`	test montrant l'affichage obtenu *test showing actual printing*
`INTFLISTE`	listes d'entiers, avec affichage modifié *lists of integers, with modified printing* hérite de / *inherits of* FLISTE
`TestIntFListe`	test montrant le dysfonctionnement de la COPIE *test showing the malfunctioning of COPyI(E)ng*
`FUtilClass`	équivalent du / *equivalent of* ALLOCATE (p, CAST=typmold)
`FLISTE2`	FLISTE corrigé avec ALLOCATE *corrected FLISTE with ALLOCATE*
`INTFLISTE2`	identique à IntFListe, sauf: héritage de FLISTE2 *identical to IntFListe, excepted for inheritance of FLISTE2*
`TESTALLOCATE`	test final de la correction / *final test of correction*

```
public class fliste {

  protected Object tete;
  protected fliste  queue;

  public fliste () {                          //constructeur.
        tete = null;
        queue= null;
  }

  public fliste (Object t, fliste q) {  //constructeur.
        tete = t;
        queue= q;
  }

  public fliste cons (Object o) {
        return new fliste (o,this);      // Probleme...
  }

  public boolean nonVide (){return queue != null;}

  protected void affcar () {
        if (tete instanceof clint)
           ((clint)tete).print();
        else
           System.out.print(tete);
  }

  public void printclass () {
    System.out.println("Je suis de la classe: "
                + getClass().getName());
  }

  public void mapcar () {
        fliste l = this;
        while (l.nonVide()) {
          l.affcar();
          l=l.queue;
        }
        System.out.println();
        printclass();
  }


 public fliste copie () {
        fliste ll = new fliste(),        //probleme...
               lsource=this, lres=ll;
        while (lsource.nonVide()) {
                ll.tete = lsource.tete;
                lsource = lsource.queue;
                ll.queue= new fliste(); //probleme...
                ll=ll.queue;
        }
        return lres;
  }
}
```

```
class ftestliste {
  public static void main (String args[]) {
        fliste ll = new fliste();
        int [] ti = {10,20,30,40,50};
        int i,j=0,max=5,k;
        for (i=0; i<max; i++) {
                ll=ll.cons(new clint(ti[i]));
        }
        ll.copie().mapcar();
  }
}
```

```
I$ java FTESTLISTE
5040302010
Je suis de la classe: fliste
I$
```

```
public class intfliste extends fliste {

  public intfliste () { super();}

  public intfliste (int it, intfliste q) {       // constructeur
        super (new clint(it), q);
  }

  public intfliste cons (int val) {
        return new intfliste (val, this);
  }

  public int Tete () { return ((clint)tete).I;}

  protected void affcar () {
        System.out.print (" " + this.Tete()  + " ");
  }
}
```

```
class testintfliste {
  public static void main (String args[]) {
        intfliste ll = new intfliste();
        int [] ti = {10,20,40,50,1,9,86,45,33,99,0,77};
        int i;
        for (i=0; i< ti.length; i++) ll=ll.cons(ti[i]);
        ll.mapcar();
        System.out.println();
        ll.copie().mapcar();
  }
}
```

```
I$ java TESTINTFLISTe
 77   0   99   33   45   86   9   1   50   40   20   10
Je suis de la classe: intfliste

770993345869150402010
Je suis de la classe: fliste
I$
```

```
public class fliste2 {

  protected Object tete;     //acces restreint au cone
  protected fliste2  queue;

  public fliste2 () {        //constructeur.
        tete = null;
        queue= null;
  }

  protected fliste2 creer () { //"selfclass" dynamique.
        return (fliste2)fUtilClass.ALLOCATE (this);
  }

  public fliste2 cons (Object o) {
        fliste2 ll = creer();
        ll.tete = o;
        ll.queue= this;
        return ll;
  }

  public boolean nonVide (){return queue != null;}

  protected void affcar () {
        if (tete instanceof clint)
           ((clint)tete).print();
        else if (tete instanceof clchar)
           ((clchar)tete).print();
        else
           System.out.print(tete);
  }

  public void printclass () {
    System.out.println("Je suis de la classe: "
                + getClass().getName());
  }

  public void mapcar () {
        fliste2 l = this;
        while (l.nonVide()) {
          l.affcar();
          l=l.queue;
        }
        System.out.println();
        printclass();
  }


  public fliste2 copie () {
        fliste2 ll = creer(), lsource=this, lres=ll;
        while (lsource.nonVide()) {
                ll.tete = lsource.tete;
                lsource = lsource.queue;
                ll.queue= creer();
                ll=ll.queue;
        }
        return lres;
  }
}
```

```
public class intfliste2 extends fliste2 {

  public intfliste2 () { super();}

  public intfliste2 cons (int val) {
        return cons(new clint(val));
  }

  public intfliste2 cons (clint premier) {
        return (intfliste2)super.cons (premier);
  }

  public int Tete () { return ((clint)tete).I;}

  protected void affcar () {
        System.out.print (" " + this.Tete()  + " ");
  }
}
```

```
public class fUtilClass {

   // "selfclass" dynamique:
   // renvoie un objet du meme type dynamique que CAST.
   // Returns an object with the dynamic type of CAST.

   public static final Object ALLOCATE (Object CAST) {
          Object clone = null;
          try{ clone = CAST.getClass().newInstance();}
          catch(IllegalAccessException e1){}
          catch(InstantiationException e2){}
          finally{ return clone;}
   }
}
```

```
class testallocate {
  public static void main (String args[]) {
        intfliste2 ll = new intfliste2();
        int [] ti = {10,20,40,50,1,9,86,45,33,99,0,77};
        int i;
        for (i=0; i< ti.length; i++) ll=ll.cons(ti[i]);
        ll.mapcar();
        System.out.println();
        ll.copie().mapcar();
  }
}
```

```
I$ java testallocate
 77   0   99   33   45   86   9   1   50   40   20   10
Je suis de la classe: intfliste2

 77   0   99   33   45   86   9   1   50   40   20   10
Je suis de la classe: intfliste2
I$
```