Generic Java

Introduction de types paramétrés en JAVA:

- plus d'informations de types
- moins de casts

$\Rightarrow$ programmes plus sûrs

Pour faire comme

- templates C++
- packages generiques en Ada95
- polymorphisme paramétrique en O'CAML ou Haskell

Caractéristiques

- sur-ensemble de JAVA
- compile vers la JVM
- utilise les bibliotheques existantes

Syntaxe

- classes paramétrées : `Nom<Parametre>`

  le parametre de type `Parametre` peut etre utilisé comme type (dans des déclarations, ou des constructions) dans la déclaration de la classe.

- interface paramétrées : idem

- classes avec plusieurs parametres : `Nom<Param1 , Param2>`

un premier exemple MyQueue.java

```
class Vide extends RuntimeException {}
class Pleine extends RuntimeException {}

class MyQueue<A> {
  int taille, longueur;
  A[] q;
  int tete, fin;

  MyQueue(int n) {taille = n; q = new A[n];}

  MyQueue(int n, A v) {
    taille = n;
    q = new A[n];
```

```
    for (int i=0; i<n; i++) {q[i] = v;}
  }

  boolean estVide() {return (longueur == 0);}

  void entrer(A x) throws Pleine {
    if (longueur < taille) { q[fin++ % taille] = x; longueur++;}
    else throw new Pleine();
  }

  A partir() throws Vide {
    if (longueur > 0) { longueur--; return q[tete++ % taille];}
    else throw new Vide();
  }
}
```

un premier exemple MainMyQueue.java

```
class MainMyQueue {
  public static void main (String [] a) {
  MyQueue<String> qs = new MyQueue<String>(10);
  qs.entrer("PREMIER");
  qs.entrer("DEUZE");
  qs.entrer("TER");
  System.out.println(qs.partir());
  System.out.println(qs.partir());
  System.out.println(qs.partir());

  MyQueue<Integer> qi = new MyQueue<Integer>(10);
  qi.entrer(new Integer(1));
  qi.entrer(new Integer(2));
```

```
  qi.entrer(new Integer(3));
  System.out.println(qi.partir());
  System.out.println(qi.partir());
  System.out.println(qi.partir());

  qs.entrer("QUATRO");
  qi.entrer(new Integer(4));
/*
  if (qi.partir() == qs.partir()) {System.err.println("ETRANGE");}

  MainMyQueue.java:24: incomparable types: java.lang.Integer
                                       and java.lang.String
  if (qi.partir() == qs.partir()) {System.err.println("ETRANGE");}
                   ^
1 error
*/
```

```
 MyQueue<MyQueue<String>> qqs = new MyQueue<MyQueue<String>>(10);
 qqs.entrer(qs);
 System.out.println(qqs.partir().partir());;
 System.out.println(qs.partir());
}}
```

un premier exemple :e xecution

```
PREMIER
DEUZE
TER
1
2
3
QUATRO
Exception in thread "main" Vide
        at MyQueue.partir(MyQueue.java)
        at MainMyQueue.main(MainMyQueue.java)
```

Interfaces

```
interface Comparator<A> {
  public int compare(A x, A y);
 }

class IntegerComparator implements Comparator<Integer> {
  public int compare (Integer x, Integer y) {
    return x.intValue() - y.intValue();
  }
}
```

exemple 2 : Collections

```
class Collections {
  public static <A> A max (MyQueue<A> xs, Comparator<A> c) {
    A w = xs.partir();
    while (!(xs.estVide()) ){
      A x = xs.partir();
      if (c.compare(w,x) < 0 ) w = x;
    }
    return w;
  }
}
```

Main

```
class MainCollections {
  public static void main (String[] a) {
    MyQueue<Integer> qb = new MyQueue<Integer>(10);
    qb.entrer(new Integer(12));
    qb.entrer(new Integer(8));
    qb.entrer(new Integer(16));
    qb.entrer(new Integer(11));
    Integer m = Collections.max(qb, new IntegerComparator());
    System.out.println(m.intValue());  // 16

   MyQueue<String> qs = new MyQueue<String>(10);
   qs.entrer("PREMIER");
/*
```

```
   String s = Collections.max(qs, new IntegerComparator());

MainCollections.java:14: method max(MyQueue<java.lang.String>,
                                    IntegerComparator)
not found in class Collections
   String s = Collections.max(qs, new IntegerComparator());
                          ^
1 error
i*/

  }
}
```

pas de casts nécessaires

- plus grande sûreté
- inférence de types pour les parametres de types :

  Le plus petit parametre de type qui permet un appel de methode valide est choisi :

  L'appel de `max` avec une queue d'Integer et un comparateur d'Integer infere que la parametre de type `A` est `Integer`

Sous-typage

- `String` est un sous-type d'`Object`
- `String[]` est un sous-type d'`Object[]`
- `Queue<String>` N'EST PAS UN SOUS-TYPE de `Queue<Object>`

Limitations

- `new A()` est interdit si `A` est un parametre de type
- `new A[n]` engendre un warning est non verifié (utiliser Vector)
- utilisation de cast sans le parametre de types

A FAIRE

1. Installer la distribution :
   `http://www.cis.unisa.edu.au/~pizza/gj`
2. Télécharger la documentation (meme URL)
3. Tester ces exemples
4. Utiliser GJ a' une structures de données d'un de vos (nombreux) programmes Java