- Package Java.util
- Classe générique

Classe Vector

•La taille est dynamique: dès qu'un tableau vectoriel est plein,
sa taille est doublée, triplée, etc. automatiquement
•Les cases sont de type Object

•add(object o), add(int i, Object o), addElement(Object o),
insertElementAt(Object o, int i) pour ajouter un objet
•remove (Object o), removeElement(Object o) removeElementAt(int i)
pour supprimer un objet
•get (int i) et elementAt(int i) retourne l'adresse de l'objet à l'indice i
•indexof(Object o) retourne la position de l'objet
•size()

```
import java.util.*;
public class UseVector1{
public static void main(String args[]){
Vector vect = new Vector(5);
vect.addElement(new Integer(3));
vect.addElement(new String("Bonjour"));
vect.addElement(new Float(3.14));
vect.addElement(new Integer(1));
vect.addElement(new Integer(2));
vect.addElement(new Integer(3));

System.out.println("après add : " + vect + "   et  taille = "
     + vect.size());
System.out.println("capacité du vecteur : " + vect.capacity());
System.out.println(vect);
for(int i=0; i < vect.size(); i++)
   if(vect.elementAt(i) != null)
   System.out.println("vect[" + i + "] : " +
   vect.elementAt(i).getClass().getName() + " --> "
   + vect.elementAt(i));
    else
    System.out.println("vect[" + i + "] est null");
    if(vect.elementAt(1) != null){vect.set(1, new Integer(1000));}
    System.out.println("après set: " + vect);
    vect.remove(0);
    System.out.println("après remove : " + vect + "
    et  taille ="   + vect.size()); }
   }
```

```
apr?s add : [3, Bonjour, 3.14, 1, 2, 3]   et  taille = 6
capacit? du vecteur : 10
[3, Bonjour, 3.14, 1, 2, 3]
vect[0] : java.lang.Integer --> 3
vect[1] : java.lang.String --> Bonjour
vect[2] : java.lang.Float --> 3.14
vect[3] : java.lang.Integer --> 1
vect[4] : java.lang.Integer --> 2
vect[5] : java.lang.Integer --> 3
apr?s set: [3, 1000, 3.14, 1, 2, 3]
apr?s remove : [1000, 3.14, 1, 2, 3]   et  taille = 5
```

Iterator

public boolean hasNext()

public Object next()

public void remove() (enlève le dernier élément retourné)

```
import java.util.*;

class Point
{private int x,y;
public Point(int a, int b)
{x=a;  y=b;}
public int getx()  {return x;}
public int gety()  {return y;}
public void affiche() {System.out.println("x = "+x+" y = "+y);}
}

class Gestion_tab
{ private Vector clt; //private Linkedkist clt;
  private int nb_pts_positifs;
public Gestion_tab( int n)
{clt= new Vector(); // clt=new linkedList();
nb_pts_positifs=0;}

public void ajoute_pt(int a, int b)
{       Point pp= new Point(a,b);
        clt.add(pp);
}
public void affiche_tous_pts()
{for (int i=0; i <clt.size(); i++)
        {Object obj=clt.get(i);
        Point pp=(Point)obj;
        pp.affiche();}
}
```

```
public void calcul_pts_positifs()
{int nb=0;
Iterator it=clt.iterator();
while (it.hasNext())
         {Object obj=it.next();
         Point pp=(Point) obj;
         if ((pp.getx() >0) &&(pp.gety()>0)) nb++;
         }
         nb_pts_positifs=nb;
         System.out.println("Nombre de points positifs= "+
         nb_pts_positifs);
}
}
public class TestVector
{ public static void main (String[] args)
         {Gestion_tab t=new Gestion_tab();
         int nb_pts=Saisie.lire_entier();
         t.ajoute_pt(2,3);
         t.ajoute_pt(5,5);
          t.affiche_tous_pts();
         t.calcul_pts_positifs();}
}
```

Classe générique

```
class Gclasse<T>
{       private T t;
        //autre variables
public Gclasse(T u)
        {t=u;}
//autre méthodes
}
```

C'est la déclaration de la classe générique de paramètre T

```
public void method1(T v) {   }

public <T extends Personne> void method2() {   }
```

(la méthode doit respecter T hérite de Personne)

```
public  <T extends Personne> T method4() {   }

public T  method5 () {  }
```

•T peut être les types prédéfinis (String, Integer,…)
ou les classes Personne, Point, …

•T est déclaré <T>

•? designe le type inconnu, il peut prendre n'importe quelle valeur de type
<? extends xxx>

•Il ne peut pas avoir de constructeur T(…)

Tableaux génériques

```
class Gestion_tab
{ private Vector<Point> clt;
   private int nb_pts_positifs;
   public Gestion_tab( )
         {clt= new Vector<Point>();
         nb_pts_positifs=0; }
public void ajoute_pts(int n)
{for (int i=0; i <n;i++)
         {Point pp= new Point();  clt.add(pp);} }
public void affiche_tous_pts()
{for (Point pp :clt)    pp.affiche();   }
public void calcul_pts_positifs()
{int nb=0;
for (Point pp:clt)   if ((pp.getx() >0) &&(pp.gety()>0)) nb++;
nb_pts_positifs=nb;
System.out.println("Nombre de points positifs= "+ nb_pts_positifs);}
}
public class TestVector1
{ public static void main (String[] args)
         {Gestion_tab t=new Gestion_tab();  int nb_pts=Saisie.lire_entier();
         t.ajoute_pts(nb_pts);   t.affiche_tous_pts();
         t.calcul_pts_positifs();}
}
```

```
class Point implements Comparable<Point>
{

public int compareTo( Point p)
{  int n=0; if (this.x < p.x)  n=-1;   if (this.x ==p.x) n=0; if (this.x >p.x) n=1;
return n;    }
}
class Gestion_tab
{ private Vector<Point> clt;
   private int nb_pts_positifs;


public void tri()
{Collections.sort(clt);}
```

```
class Cl_Obj
{private Object t;

public Cl_Obj(Object u)
{t=u;}

public Object get()
{return t;}
}

public class Testobjet
{ public static void main(String[] args)
        {Cl_Obj nb1= new Cl_Obj (new Integer(37));
        Cl_Obj st1= new Cl_Obj("bonjour");
        int  v=(Integer)nb1.get();//pas int;
        String s=(String)st1.get();
        System.out.println(v);
        System.out.println(s);
        nb1=st1;
        v=(Integer)nb1.get(); // erreur l'exécution
}}
```