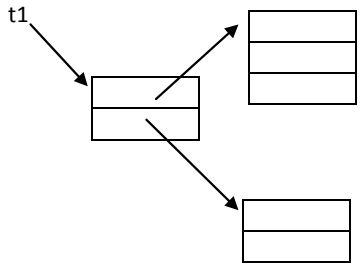


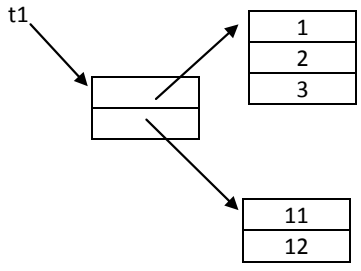
— la notation t1[0] désigne la référence au premier tableau de 3 entiers,
— la notation t1[0][1] désigne le deuxième élément de ce tableau (les indices commencent à 0),
— la notation t1[1] désigne la référence au second tableau de 2 entiers,
— la notation t1[1][i-1] désigne le i_{me} élément de ce tableau,
— l'expression t1.length vaut 2,
— l'expression t1[0].length vaut 3,
— l'expression t1[1].length vaut 2.

2.3.2 Initialisation

Dans le premier exemple, nous avons utilisé un initialiseur pour les deux références à introduire dans le tableau t1 ; autrement dit, nous avons procédé comme pour un tableau à un seul indice. Mais comme on s'y attend, les initialiseurs peuvent tout à fait s'imbriquer, comme dans cet exemple :

```
int t[] []={{1,2,3}, {11,12}};
```

La situation précédente correspond à schéma suivant :



2.4 Exercices

2.4.1 Déclaration et initialisation de tableau

Quelles erreurs ont été commises dans la méthode test ?

```
public void Application ()
{
  public static void main (String[] args)
   {
        int i=20 ;
        final int ii=50;
        int ti[]={1,3,5};
        int ti2[]={i+10,i,i+10};
        int ti3[]={ii-10,ii,ii+11};
        int ti4[];
        ti4={11,13,59};
        float tf1[]={12,22,p,p+21} ;
        float tf2[]={1.85,2.25,52} ;
        double td[]={1,22.5,25.25,29*p};
   }
}
```

2.4.2 Affectation de tableaux

Que se passera-t-il si nous exécutons la méthode test ?

```
public class Tableaux
{
  public void test()
   {
        int ti1[]={11,12,13};
        int ti2[]=new int[4];
        for(int i=0;i<4;i++) ti2[i]=2*i;
          ti2=ti1;
        for(int i=0;i<4;i++) System.out.println(ti2[i]);
   }
}
```

2.4.3 Affectation de tableaux(2)

Quels résultats fournit la méthode test ?

```
public class Tableaux
{
  public void test()
        { char tc1[]={'b','o','n','j','o','u','r'};
          char tc2[]={'h','e','l','l','o'};
          char tc3[]={'x','x','x','x'};
          tc3=tc1; tc1=tc2; tc2=tc3;
          System.out.print("tc1=␣");
          for(int i=0;i<tc1.length;i++) System.out.print(tc1[i]);
          System.out.println();
          System.out.print("tc2=␣");
          for (int i=0;i<tc2.length;i++) System.out.print(tc2[i]);
       System.out.println();
       System.out.print ("tc3␣=␣");
          for (int i=0 ; i<tc3.length ; i++) System.out.print (tc3[i]);
          System.out.println();
    }
}
```

2.4.4 Tableau en argument

Écrire une classe UtiliserTableaux disposant des méthodes statiques suivantes :
— *sommerTab* qui fournit la somme des valeurs d'un tableau de réels (double) de taille quelconque.
— *incrementerTab* qui incrémente d'une valeur donnée toutes les valeurs d'un tableau de réels (double).
— *afficherTab* qui affichage les valeurs d'un tableau de réels.

2.4.5 Tableau en valeur de retour

Écrire une classe UtiliserTableaux2 disposant des méthodes statiques suivantes :
— genererTab qui fournit en retour un tableau des n premiers nombres impairs, la valeur de n étant fournie en argument
— sommerTab qui reçoit en argument deux vecteurs d'entiers de même taille et qui fournit en retour un tableau représentant la somme de ces deux vecteurs.

2.4.6 Tableaux de tableaux

Quels résultats fournit le méthode test ?

```
public class TabDeTableaux
{
  public void test ()
        {
          int[][] m=new int[3][];
          for(int i=0;i<3;i++)
           {
            m[i]=new int[i+1];
                for(int j=0;j<m[i].length;j++)
                        m[i][j]=i+j ;
       }
          for(int i=0;i<3;i++)
           {
             System.out.print ("tableau numero "+i+" = ") ;
             for (int j=0;j<m[i].length;j++)
                 System.out.print (m[i][j]+" ") ;
                 System.out.println();
       }
     }
}
```