Corrigé du TP de Java n°3

1 CE NOMBRE EST-IL PREMIER ?

```
int a=59;
int diviseur=2;
int nest_pas_premier=0;
while(diviseur < a)
{
	if(a%diviseur == 0)
	{
		nest_pas_premier=1;
	}
	diviseur=diviseur+1;
}
if(nest_pas_premier == 1)
{
	System.out.println("Pas premier");
}
else
{
	System.out.println("Premier");
}
```

2 MAIS QUI A GAGNÉ LES ÉLECTIONS ?

```
import java.util.*;
import java.io.*;

public class Main
{
	public static void main(String args[])
	{
		int[] result_vote = new int[10017];
		int i=0;
		try
		{
			FileInputStream fstream = new
FileInputStream("resultat_vote.txt");
			DataInputStream in = new DataInputStream(fstream);
			BufferedReader br = new BufferedReader(new
InputStreamReader(in));
			String strLine;
			while ((strLine = br.readLine()) != null)
			{
```

```
                        result_vote[i]=Integer.parseInt(strLine);
                        i++;
                  }
                  in.close();

            }
            catch (Exception e)
            {
                        System.err.println("Error: " + e.getMessage());
            }

            //Insérez votre code ici...
            int compteur, vote_chuck, vote_bruce;
            vote_chuck=0;
            vote_bruce=0;
            for(compteur=0; compteur<result_vote.length; compteur=compteur+1)
            {
                  if(result_vote[compteur] == 110)
                  {
                        vote_chuck=vote_chuck+1;
                  }
                  else if(result_vote[compteur] == 101)
                  {
                        vote_bruce=vote_bruce+1;
                  }
            }

            if(vote_chuck>vote_bruce)
            {
                  System.out.println("Norris president");
            }
            else if(vote_bruce>vote_chuck)
            {
                  System.out.println("Willis president");
            }
            else
            {
                  System.out.println("Guerre civile !!!!");
            }
      }
}
```

3 TRIER UN TABLEAU

Pour trier un tableau, on va parcourir le tableau case par case, et examiner les cases par paires voisines : si la case à la position j contient une donnée plus grande que la case à la position j+1, alors on échangera les valeurs de deux cases. Effectuer cette simple opération ne suffit pas à trier le tableau : il faut, une fois deux valeurs permutées, recommencer à parcourir le tableau depuis le début.

```
import java.util.Random;

class Main
{
      public static void main(String args[])
      {
            Random m = new Random();
```

```
            //Construction du premier tableau
            int hasard = m.nextInt(20)+1;
            int tab1[];
            tab1 = new int[hasard];
            int x=0;
            while(x<tab1.length)
            {
                  hasard = m.nextInt(101);
                  tab1[x] = hasard;
                  x=x+1;
            }

            int compteur=0;
            while(compteur<tab1.length-1)
            {
                  if(tab1[compteur]>tab1[compteur+1])
                  {
                        t=tab1[compteur];
                        tab1[compteur]=tab1[compteur+1];
                        tab1[compteur+1]=t;
                        compteur=-1; //Une fois deux cases permutées, on repart
au début du tableau
                  }
                  compteur=compteur+1;
            }

            x=0;
            while(x<tab1.length)
            {
                  System.out.println(tab1[x]);
                  x=x+1;
            }
      }
}
```

Cette algorithme fonctionne mais est long... Si on trie un tableau de 10000 cases, cela prendre 10 minutes ~ 15 minutes pour tout trier (une fois le tri effectué, vous ne verrez pas grand chose car le terminal ne peut pas afficher toutes les cases du tableau)...

Testez en modifiant la ligne

```
tab1 = new int[hasard];
```

par

```
tab1 = new int[10000];
```

Pour résoudre le problème, on va faire comme suit : quand deux cases sont permutées, on laisse le programme finir de parcourir le tableau. Puis, on recommence la même séquence d'actions (balayer le tableau du début à la fin et échanger les valeurs mal positionnées). On recommencera cette opération autant de fois qu'il y a de cases dans le tableau : ceci garantit de trier le tableau, et de façon plus rapide que le programme précédent.

```
import java.util.Random;

class Main
{
      public static void main(String args[])
      {
            Random m = new Random();
```

```
            //Construction du premier tableau
            int hasard = m.nextInt(20)+1;
            int tab1[];
            tab1 = new int[hasard];
            int x=0;
            while(x<tab1.length)
            {
                  hasard = m.nextInt(101);
                  tab1[x] = hasard;
                  x=x+1;
            }

            int z=0;
            for(z=0; z<tab1.length; z=z+1)
            {
                  compteur=0;
                  while(compteur<tab1.length-1)
                  {
                        if(tab1[compteur]>tab1[compteur+1])
                        {
                              t=tab1[compteur];
                              tab1[compteur]=tab1[compteur+1];
                              tab1[compteur+1]=t;
                              compteur=-1; //Une fois deux cases permutées, on
repart au début du tableau
                        }
                        compteur=compteur+1;
                  }
            }

            x=0;
            while(x<tab1.length)
            {
                  System.out.println(tab1[x]);
                  x=x+1;
            }
      }
}
```

4 DÉCOMPOSITION EN NOMBRE PREMIER

```
int a=59;
int copie_a=a;
int diviseur=2;
while(copie_a!=1)
{
      if(copie_a%diviseur == 0)
      {
            System.out.println(diviseur);
            copie_a=copie_a/diviseur;
      }
      else
      {
            diviseur=diviseur+1;
      }
}
```