Vergleich von drei Sortierverfahren

Bubblesort

```
public void bubblesort(int [] a)
{
   int temp;

   for (int i=0; i<a.length; i++)
      for (int j=0; j<a.length-i-1; j++)
         if (a[i]>a[i+1])
         {
            temp = a[i];a[i]=a[i+1];a[i+1]=temp;
         }
}
```

Mittelwerte von jeweils 15 Durchgängen:

```
Sortierzeit für     1000 Zahlen =   0,80 Millisekunden
Sortierzeit für     2000 Zahlen =   0,80 Millisekunden
Sortierzeit für     4000 Zahlen =   3,13 Millisekunden
Sortierzeit für     8000 Zahlen =  11,80 Millisekunden
Sortierzeit für    16000 Zahlen =  38,67 Millisekunden
Sortierzeit für    32000 Zahlen = 154,67 Millisekunden
Sortierzeit für    64000 Zahlen = 621,40 Millisekunden
Sortierzeit für   128000 Zahlen = 2463,33 Millisekunden
```

Insertionsort

```
public void insertionsort(int [] a)
{
    int j, temp;

    for (int i = 1; i < a.length; i++)
    {
        j = i;
        temp = a[i];

        while ((j > 0) && (a[j-1]>temp))
        {
            a[j] = a[j-1];
            j--;
        }
        a[j] = temp;
    }
}
```

Mittelwerte von jeweils 15 Durchgängen:

```
Sortierzeit für     1000 Zahlen =   0,87 Millisekunden
Sortierzeit für     2000 Zahlen =   0,67 Millisekunden
Sortierzeit für     4000 Zahlen =   2,73 Millisekunden
Sortierzeit für     8000 Zahlen =   9,40 Millisekunden
Sortierzeit für    16000 Zahlen =  39,20 Millisekunden
Sortierzeit für    32000 Zahlen = 158,20 Millisekunden
Sortierzeit für    64000 Zahlen = 636,73 Millisekunden
Sortierzeit für   128000 Zahlen = 2575,47 Millisekunden
```

Mergesort

```
public void merge
(int[] a, int[] l, int[] r, int left, int right)
{
    int i = 0, j = 0, k = 0;
    while (i < left && j < right)
    {
        if (l[i] <= r[j])
            a[k++] = l[i++];
        else
            a[k++] = r[j++];
    }
    while (i < left)
        a[k++] = l[i++];
    while (j < right)
        a[k++] = r[j++];
}

public void mergeSort(int[] a, int n)
{
    if (n < 2) return;

    int mid = n / 2;
    int[] l = new int[mid];
    int[] r = new int[n - mid];

    for (int i = 0; i < mid; i++)
        l[i] = a[i];
    for (int i = mid; i < n; i++)
        r[i - mid] = a[i];

    mergeSort(l, mid);
    mergeSort(r, n - mid);

    merge(a, l, r, mid, n - mid);
}
```

Mittelwerte von jeweils 15 Durchgängen:

```
Sortierzeit für     1000 Zahlen =   0,40 Millisekunden
Sortierzeit für     2000 Zahlen =   0,47 Millisekunden
Sortierzeit für     4000 Zahlen =   1,13 Millisekunden
Sortierzeit für     8000 Zahlen =   2,27 Millisekunden
Sortierzeit für    16000 Zahlen =   4,00 Millisekunden
Sortierzeit für    32000 Zahlen =   7,07 Millisekunden
Sortierzeit für    64000 Zahlen =  11,27 Millisekunden
Sortierzeit für   128000 Zahlen =  23,13 Millisekunden
Sortierzeit für   256000 Zahlen =  49,00 Millisekunden
Sortierzeit für   512000 Zahlen = 110,47 Millisekunden
Sortierzeit für  1024000 Zahlen = 218,80 Millisekunden
```

Konstruktor für alle drei Verfahren:

```
public Sort()
{
   for (int max = 1000; max < 2000000; max = max*2)
   {
      double ms = 0;
      for (int i=1; i<=15; i++)
      {
        int[] liste = ArrayTools.createRandom(max,2);
        final long timeStart = System.currentTimeMillis();
        sort(liste);
        final long time = System.currentTimeMillis() - timeStart;
        ms += time;
      }
      System.out.printf("Sortierzeit für %8d Zahlen = %6.2f
                         Millisekunden\n",max,ms/15);
   }
}
```

Für Bubblesort und Insertionsort wurde max auf 200.000 begrenzt, weil das Sortieren sonst ewig gedauert hätte.

Aufgaben:

1. Implementieren und testen Sie den Mergesort!
2. Erstellen Sie mit OpenOffice eine Präsentation, die den Mergesort erklärt!