Aufgabenblatt: Arrays

(1.) (a.) Erstellen Sie eine Methode, die in einem Array von Zahlen nach einem bestimmten Wert sucht!

```
static int LinearSearch(int searchValue, int Array2Search[]) {

}
```

(b.) Erstellen Sie eine Methode LinearSearchString, die nach einer Zeichenkette innerhalb eines Arrays von Zeichenketten sucht! Verwenden Sie nicht zum Vergleich „==“, sondern equals!
(c.) Erstellen Sie eine Methode LinearSearchFrom, die in einem Array von Zahlen nach einem bestimmten Wert ab einer bestimmten Stelle sucht!
(d.) Erstellen Sie eine Methode LinearSearchXth, die in einem Array von Zahlen nach einem bestimmten Wert nach dem X-ten Vorkommen sucht!

(2.) Googlen Sie nach binärer Suche und kopieren Sie eine Methode für binäres Suchen in ihren Quellkode und passen Sie diesen an!
(a.) iterativ
(b.) rekursiv

Wenn Sie nicht fündig werden, können Sie mit diesem Quellkode

```
int[] data;
int size;

public boolean binarySearch(int key)
{
    int low = 0;
    int high = size - 1;

    while(high >= low) {
       int middle = (low + high) / 2;
       if(data[middle] == key) {
          return true;
       }
       if(data[middle] < key) {
          low = middle + 1;
       }
       if(data[middle] > key) {
          high = middle - 1;
       }
    }
    return false;
}
```

(3.) Nach Eingabe eines dreistelligen Integerwertes soll dessen Index mittels der iterativen Funktion aus Aufgabe 2 im Array ermittelt und ausgegeben werden. Falls der dreistellige Integerwert nicht existiert, so muss ein entsprechender Hinweis ausgegeben werden:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Inhalt	112	131	140	199	201	209	332	345	588	666	723	799	810	889	899

(a.) Suchen Sie mit der Methode binarySearch nach 201, 131, 799 und 899 und geben Sie die Werte in folgender Tabelle an:

Durchlauf	*low*	*middle*	*high*
1			
2			
3			
4			

(b.) Wie viele Suchschritte müssen für ein Array für n Zahlen realisiert werden?

n	*Suchschritte lineare Suche*	*Suchschritte binäre Suche*
10		
100		
1000		
1.000.000		
1.000.000.000		

(c.) Stellen Sie das Programm als Struktogramm dar!

(4.) Sortieren Sie das Array 21, 45, 11, 18, 50, 32 mit allen bekannten Sortierverfahren und vergleichen Sie die Schritte/Ergebnisse mit dem Nachbarn!
Verwenden Sie zur Kontrolle die Seite:
https://visualgo.net/sorting
Kopieren Sie die Daten und führen Sie die Sortierung nach allen Sortierverfahren durch!

(5.) Sortieren Sie die Zahlenfolge „35 28 41 7 14 50 33 21 21 60 18 12“ und dokumentieren Sie die einzelnen Schritte!

(6.) (a.) Erstellen Sie ein Programm, welches nacheinander mit BubbleSort, QuickSort und ShellSort ein Array von Zufallszahlen sortiert! Quellcodes in Javascript hier:
http://www.w3resource.com/javascript-exercises/searching-and-sorting-algorithm/index.php
(b.) Messen Sie die Zeit der einzelnen Algorithmen für 1.000.000 Elemente!
(c.) Packen Sie die Sortieralgorithmen in eine Klasse CSort und die Zeitmessungsmethoden in eine Klasse CTimer

Aufgabenblatt: Arrays

(1.) (a.)

```
public class Test  {

  static int LinearSearch(int searchValue, int Array2Search[]) {
    for (int i=0;i<Array2Search.length;i++) {
      if (searchValue==Array2Search[i]) {
        return i+1;
      } // end of if
    }  // end of for
    return -1;
  }

  public static void main(String[] args) {
    int Data[]={1,3,5,-4};

    int value=3;
    int foundAt=LinearSearch(value,Data);
    if (foundAt!=-1)
      System.out.print("Wert "+value+" an der Stelle "+foundAt+" gefunden!");
    else
       System.out.print("Wert "+value+" nicht gefunden!");
  } // end of main
}
```

(b.)

```
public class Test  {

  static int LinearSearchString(String searchValue, String Array2Search[]) {
    for (int i=0;i<Array2Search.length;i++) {
      if (searchValue==Array2Search[i]) {
        return i+1;
      } // end of if
    }  // end of for
    return -1;
  }

  public static void main(String[] args) {
    String  Data[]={"Peter","Paul","und","Hava", "programmieren","alle","Java"};

    String value="Hava";
    int foundAt=LinearSearchString(value,Data);
    if (foundAt!=-1)
    System.out.print("Text "+value+" an der Stelle "+foundAt+" gefunden!");
    else
    System.out.print("Text "+value+" nicht gefunden!");
  } // end of main
}
```

(c.)

```
public class Test  {

  static int LinearSearchFrom(int searchValue, int Array2Search[],int startPos) {
    for (int i=startPos;i<Array2Search.length;i++) {
      if (searchValue==Array2Search[i]) {
        return i+1;
      } // end of if
    }  // end of for
    return -1;
  }

  public static void main(String[] args) {
    int Data[]={1,3,5,-4};

    int value=5;
    int foundAt=LinearSearchFrom(value,Data,2);
    if (foundAt!=-1)
    System.out.print("Wert "+value+" an der Stelle "+foundAt+" gefunden!");
    else
    System.out.print("Wert "+value+" nicht gefunden!");
  } // end of main
}
```

(d.)

```
public class Test  {

  static int LinearSearchXth(int searchValue, int Array2Search[], int Xth) {
    int NoOfFounds=0;
    for (int i=0;i<Array2Search.length;i++) {
      if (searchValue==Array2Search[i]) {
        NoOfFounds++;
        if (NoOfFounds==Xth)
        return i+1;
      } // end of if
    }  // end of for
    return -1;
  }

  public static void main(String[] args) {
    int Data[]={1,3,5,-4,3,4,2,3,1};

    int value=3;
    int Xth=2;
    int foundAt=LinearSearchXth(value,Data,Xth);
    if (foundAt!=-1)
    System.out.print("Wert "+value+" an der Stelle "+foundAt+" zum "+Xth+"-ten Mal gefunden!");
    else
    System.out.print("Wert "+value+" nicht gefunden!");
  } // end of main
}
```

(2.)

```
public class Test  {

   public static int binarySearch(int searchValue, int Array2Search[])
   {
       int low = 0;
       int high = Array2Search.length - 1;

       while(high >= low) {
          int middle = (low + high) / 2;
          if(Array2Search[middle] == searchValue) {
             return middle;
          }
          if(Array2Search[middle] < searchValue) {
             low = middle + 1;
          }
          if(Array2Search[middle] > searchValue) {
             high = middle - 1;
          }
       }
       return -1;
   }

  public static void main(String[] args) {
    int Data[]={1,3,5,-4};

    int value=1;
    int foundAt=binarySearch(value,Data);
    if (foundAt!=-1)
      System.out.print("Wert "+value+" an der Stelle "+(int)(foundAt+1)+" gefunden!");
    else
       System.out.print("Wert "+value+" nicht gefunden!");
  } // end of main
}
```

(3.)(a.)

a) 201

Durchlauf	low	middle	high
1	0	7	14
2	0	3	6
3	3	4	5
4	4	4	4

b) 131

Durchla	low	middle	high
1	0	7	14
2	0	3	6
3	0	1	2
4	1	1	1

c) 799

Durchlauf	low	middle	high
1	0	7	14
2	7	10	13
3	10	11	12
4	11	11	11

d) 899

Durchla	low	middle	high
1	0	7	14
2	8	11	14
3	12	13	14
4	14	14	14

Den Wert von 201 durch die anderen ersetzen und das Programm laufen lassen: Die drei Werte werden ausgegeben; man sollte aber auch mit F7 mal das Programm durchlaufen lassen!

```
public class Test  {

 public static int binarySearch(int searchValue, int Array2Search[])
 {
  int low = 0;
  int high = Array2Search.length - 1;

  while(high >= low) {
   int middle = (low + high) / 2;
   if(Array2Search[middle] == searchValue) {
    return middle;
   }
   if(Array2Search[middle] < searchValue) {
    low = middle + 1;
   }
   if(Array2Search[middle] > searchValue) {
    high = middle - 1;
   }
   System.out.println("Low:"+low+" middle:"+middle+" high:"+high);
  }
  return -1;
 }

 public static void main(String[] args) {
  int Data[]={112,131,140,199,201,332,345,588,666,723,799,810,889,899};

  int value=201;
  int foundAt=binarySearch(value,Data);
  if (foundAt!=-1)
  System.out.print("Wert "+value+" an der Stelle "+(int)(foundAt+1)+" gefunden!");
  else
  System.out.print("Wert "+value+" nicht gefunden!");
 } // end of main
}
```

(b.)

n	*Suchschritte lineare Suche*	*Suchschritte binäre Suche*
10	*(1+10)/2*	$log_2 10=$
100	*(1+100)/2*	$log_2 100=$
1000	*(1+1000)/2*	$log_2 1000=$
1.000.000	*(1+1000000)/2*	$log_2 1.000.000=$
1.000.000.000	*(1+1.000.000.000)/2*	$log_2 1.000.000.000=$

(4.) trivial
(5.)

```
import java.util.Random;
public class Sortierverfahren
{
  public static void main(String args[])
  {
    int[] anArray;
    anArray=createArrayWithRandomNumbers();
    arrayToScreen(anArray);
    //quicksort(anArray,0,anArray.length-1);
    //bubbleSort(anArray);
    shellsort(anArray,anArray.length);
    arrayToScreen(anArray);
  }

  public static int[] createArrayWithRandomNumbers() {
    int[] anArray;
    Random rand = new Random();
    anArray = new int[20];
    for(int i=0;i<anArray.length;i++)
       anArray[i] = rand.nextInt(20);
    return anArray;
  }

  public static void arrayToScreen(int numbers[]) {
    for(int i=0;i<numbers.length;i++)
       System.out.print(numbers[i]+" ");
    System.out.println();
    return;
  }

 private static void quicksort(int numbers[],int low, int high) {
     int i = low, j = high;
     int pivot = numbers[low + (high-low)/2];
     while (i <= j) {
        while (numbers[i] < pivot) {
           i++;
        }
        while (numbers[j] > pivot) {
           j--;
        }
        if (i <= j) {
           int temp = numbers[i];
           numbers[i] = numbers[j];
           numbers[j] = temp;
           i++;
           j--;
        }
     }
     if (low < j)
        quicksort(numbers,low, j);
     if (i < high)
        quicksort(numbers,i, high);
  }
  public static void bubbleSort(int[] x) {
    boolean unsortiert=true;
    int temp;

    while (unsortiert){
```

```
      unsortiert = false;
      for (int i=0; i < x.length-1; i++)
        if (x[i] > x[i+1]) {
          temp       = x[i];
          x[i]       = x[i+1];
          x[i+1]     = temp;
          unsortiert = true;
        }
    }
  }

  public static void  shellsort (int[]  a, int n)
  {
     int i, j, k, h, t;

     int[] spalten = {2147483647, 1131376761, 410151271, 157840433,
     58548857, 21521774, 8810089, 3501671, 1355339, 543749, 213331,
     84801, 27901, 11969, 4711, 1968, 815, 271, 111, 41, 13, 4, 1};

     for (k = 0; k < spalten.length; k++)
     {
        h = spalten[k];
        // Sortiere die "Spalten" mit Insertionsort
        for (i = h; i < n; i++)
        {
           t = a[i];
           j = i;
           while (j >= h && a[j-h] > t)
           {
              a[j] = a[j-h];
              j = j - h;
           }
           a[j] = t;
        }
     }
  }

} // end class
```

(b.) Nur für BubbleSort, die anderen analog: Ferner habe ich aus Zeitgründen nur für 20 Elemente gecodet:

```
import java.util.Random;
public class Sortierverfahren
{
  public static void main(String args[])
  {
    int[] anArray;
    anArray=createArrayWithRandomNumbers();
    long startTime = System.nanoTime();
    bubbleSort(anArray);
    long endTime = System.nanoTime();
    long duration = (endTime - startTime);
    // In Millisekunden wäre durch 10^6 geteilt:
    // long duration = (endTime - startTime)/1000000;
    System.out.println("Zeit in ns:"+duration);
  }

  public static int[] createArrayWithRandomNumbers() {
    int[] anArray;
    Random rand = new Random();
    anArray = new int[20];
    for(int i=0;i<anArray.length;i++)
       anArray[i] = rand.nextInt(20);
    return anArray;
  }

  public static void bubbleSort(int[] x) {
    boolean unsortiert=true;
    int temp;

    while (unsortiert){
      unsortiert = false;
      for (int i=0; i < x.length-1; i++)
        if (x[i] > x[i+1]) {
          temp      = x[i];
          x[i]      = x[i+1];
          x[i+1]    = temp;
          unsortiert = true;
        }
    }
  }
```

(d.) 1. Datei CTimer.java:

```
public class CTimer {
  private long startTime;
  private long endTime;

  public void end() {
       endTime = System.nanoTime();
  }

  public void start() {
       startTime = System.nanoTime();
  }

  public long duration() {
       return
            endTime-startTime;
  }
} // end of CTimer
```

2. Datei CSort.java:

```
import java.util.Random;
public class CSort {
   // Erweiterbar durch die anderen 3 Verfahren!
   public static void bubbleSort(int[] x) {
     boolean unsortiert=true;
     int temp;

     while (unsortiert){
       unsortiert = false;
       for (int i=0; i < x.length-1; i++)
         if (x[i] > x[i+1]) {
           temp       = x[i];
           x[i]       = x[i+1];
           x[i+1]     = temp;
           unsortiert = true;
         }
     }
   }
   public static int[] createArrayWithRandomNumbers() {
     int[] anArray;
     Random rand = new Random();
     anArray = new int[20];
     for(int i=0;i<anArray.length;i++)
        anArray[i] = rand.nextInt(20);
     return anArray;
   }
} // end of CSort
```

3. Datei Sortierverfahren.java:

```
public class Sortierverfahren
{
  public static void main(String args[])
  {
     CSort mySort = new CSort();
     CTimer myTimer = new CTimer();
     int[] anArray;
     anArray=mySort.createArrayWithRandomNumbers();
     myTimer.start();
     mySort.bubbleSort(anArray);
     myTimer.end();
     System.out.println("Zeit in ns:"+myTimer.duration());
  }

}  // end class
```