Ein Programm zur binären Suche

Arbeite mit dem folgenden Programmgerüst:

App.java

```
/**
 * Erzeugt eine geordnete Zufallsreihe und ermöglicht Abfragen darüber.
 *
 * @author Frank Schiebel
 * @version 1.0
 */
class BinarySearch
{
    private int[] daten;
    int anzahl;

    public BinarySearch(int anzahl)
    {
        this.anzahl = anzahl;
        daten = new int[anzahl];
        int indexvorher = 0;
        for (int i = 0; i < daten.length; i++)
        {
            if ( i>0 ) {
                indexvorher = i -1;
            }
            daten[i] =
getZufallszahlOrdered(daten[indexvorher],10*anzahl);
        }
    }


    public int binaereSuche(int zahl) {


    }




    public void anzeigen() {
        for (int i=0; i< anzahl; i++) {
             System.out.println( i + " -> " + daten[i] + " ");
        }
    }

    private int getZufallszahlOrdered(int basis, int grenze)
    {
```

```
        return (int)(2*(grenze-basis)/anzahl*Math.random()+1) + basis;
    }

}


/* App Klasse. Steuerklasse für unser Programm */
public class App {

    public static void main(String[] args) {
        BinarySearch liste = new BinarySearch(1000);
        liste.anzeigen();


    }

}
```

Aufgaben:

A1

Probiere das Programm aus. Beschreibe, was es macht, verändere auch den Parameter `anzahl` bei der Erzeugung des BirarySearch Objekts. Welchen Einflüsse hat diese Änderung?

A2

Implementiere eine Methode `binaereSuche`, welche den Index des gesuchten Elements zurückgibt oder `-1`, wenn der gesuchte Wert nicht gefunden wird.

```
    // in der main Methode der App Klasse
    int gesucht=22;
    int treffer = liste.binaereSuche(gesucht);
    System.out.println("Der Wert " + gesucht + " befindet sich im
Arrayelement mit dem Index " + treffer);
```

- Die Methode `binaereSuche` arbeitet auf dem zuvor erzeugten Array und nimmt als Argument die gesuchte Zahl entgegen.
- Du führst Buch welcher Teil des Arrays zu durchsuchen ist und welcher Teil des Arrays nicht mehr in Frage kommt. wenn deine Methode startet, musst du das gesamte Array betrachten.

- Jetzt musst du den Index des mittleren Elements finden, und prüfen, ob der Inhalt größer,

kleiner oder gleich dem gesuchten Wert ist.

```
  int mitte = (oben+unten)/2; //Wenn oben+unten ungerade ist, wird ''mitte'' abgerundet
```

- Ist der Wert des Arrayelements **gleich** dem gesuchten Wert, wird der Index mit `return` zurückgegeben.
- Wenn der gesuchte Wert **kleiner** ist als der Inhalt von `daten[mitte]`, kann die obere Hälfte des Arrays ausgeschlossen werden, indem man als neue obere Grenze `mitte` festlegt.

```
if ( gesucht < daten[mitte]) {
   oben = mitte;
}
```

- Sollte der gesuchte Wert **größer** sein als `daten[mitte]` kann die untere Hälfte ausgeschlossen werden, dazu muss der Wert von `unten` auf `mitte` gesetzt werden.

```
if ( gesucht > daten[mitte]) {
   unten = mitte;
}
```