Onsdag den 21. april

Til brug for opgaverne 1,2,3,5 og 6 kan du anvende de to klasser, kaldet Soegning og SoegningApp i jar-filen *soegning.*

SoegningApp indeholder en main()-metode der afprøver de metoder, der laves i Soegning. Alle de nedenfor nævnte metoder laves i Soegning.

Opgave 1

Skriv en metode i Soegning, der givet et array af heltal, afgør om der findes mindst et ulige tal i *arrayet*. Skriv metoden som en søgning. Test metoden fra main() i SoegningApp.

Opgave 2

Skriv en metode i Soegning, der ved lineær søgning finder det første tal i et array af heltal, der ligger i intervallet [10,15]. Hvis arrayet ikke indeholder et sådant tal returneres -1.

Hvis arrayet ser ud som nedenstående, skal metoden altså returnere 14.

[7, 56, 34, 3, 7, 14, 13, 4]. Test metoden fra main() i AfproevSoegning.

Opgave 3

Skriv en metode i Soegning, der afgør, om der i et array står 2 ens tal ved siden af hinanden.

For	7 9 13 7 9 13	skal metoden returnere False, og
for	7 9 13 13 9 7	skal algoritmen returnere True

Skriv metoden, som en søgning. Test metoden fra main() i SoegningApp.

Opgave 4

Givet nedenstående klasse Spiller (findes i den udleverede jar fil)

```
public class Spiller {
  private String navn;
  private int hojde;
  private int vaegt;
  private int antalMaal;
    // Opretter et Spiller objekt med navn, højde, vægt og
    // antal mål
    public Spiller(String navn, int hoejde, int vaegt, int
                   antalmaal){
    this.hojde = hoejde;
    this.navn = navn;
    this.vaegt = vaegt;
    this.antalMaal = antalmaal;
  }

  // returnerer spillerens navn
  public String getNavn() {return navn;}
```

```
    // returnerer spillerens højde i cm
    public int getHøjde() {return hojde;}

    // returnerer spillerens vægt i kg
    public int getVægt() {return vaegt;}

    // returnerer antal mål spilleren har scoret i
    // indeværende sæson
    public int getMaal() {return antalMaal;}
}
```

4.1 Lav en metode der givet en ArrayList list indeholdende Spiller objekter, finder en spiller med en bestemt målscore, hvis en sådan ikke findes returneres null. Listen er ikke sorteret. Metoden skal have nedenstående signatur (en klasse med metode signaturen, kan findes i klassen *Spillemetoder* i den udleverede jar fil):

```
public Spiller findScoreLineær (ArrayList<Spiller>
spillere, int score)
```

Metoden skal laves ved lineær søgning.

4.2 Lav igen en metode der givet en ArrayList list indeholdende Spiller objekter, finder en spiller med en bestemt målscore, hvis en sådan ikke findes returneres null. Det kan antages listen er sorteret i aftagende orden efter målscore. Metoden skal have nedenstående signatur:

```
public Spiller findScoreBinær (ArrayList<Spiller> spillere,
int score)
```

Metoden skal laves ved binær søgning.

4.3 Lav en metode der givet en ArrayList list indeholdende Spiller objekter, finder navnet på en Spiller, der er mindre end 170 cm og har scoret mere end 50 mål. Listen er ikke sorteret. Metoden skal have nedenstående signatur:

```
public String godSpiller (ArrayList<Spiller> spillere)
```

Metoden skal laves ved lineær søgning. Hvis ingen spiller opfylder kriteriet returneres den tomme String.

Opgave 5

Skriv en metode, der (uden brug af operationen for kvadratrod: Math.sqrt) kan beregne heltalskvadratroden af et heltal n≥0. Metoden skal altså returnere det største heltal r, der opfylder: $r^2 \leq n < (r+1)^2$

Her er nogle eksempler:

n	r
0	0
1	1
3	1
4	2
7	2
8	2
9	3
10	3
111	10

Redegør også for, hvorledes de fem punkter fra søgeskabelonen er realiseret, og hvad der opfattes som kandidatmængde og søgemængde.

Skriv en metode der anvender **lineær** søgning

Skriv en metode der anvender **binær** søgning

Hint: der skal hverken anvendes array eller ArrayList, der skal blot søges blandt tallene 0 til n.

Opgave 6

I denne opgave betragtes en ArrayList til opbevaring af tal, der antages intet om sortering af tallene.

Skriv en metode, der finder et tal, n i en ArrayList. Metoden skal aflevere -1, hvis tallet ikke findes. Hvis tallet findes i listen, skal metoden bytte det fundne tal med tallet én position til venstre (mod mindre positioner), med mindre det allerede står på position 0. Metoden skal så aflevere den nye position. Eksempel:

Listen før søgning efter 13

6	4	8	13	2

listen efter søgning efter 13

6	4	13	8	2

og resultatet af metoden er 2.

```
public int find(ArrayList<Integer> list, int n)
```

```
/**
  * returner positionen på n efter det evt. er flyttet én position til venstre. Hvis n ikke findes
  * returneres -1.
 **/
```