HÖGSKOLAN
I HALMSTAD

Verónica Gaspes
ITE
16 november 2023

Inlämningsuppgifter för vecka 4

Anmärkningar

- Hur man lämnar in och vad som ska lämnas in förklaras i Blackboard. Denna dokument beskriver uppgifterna. Varje uppgift finns i en .java fil. I varje .java fil finns en kommentar som förklarar uppgifterna på engelska.
- När du löser veckans uppgifter behöver du bara använda det vi har presenterat under första, andra, tredje och fjärde veckorna:
  - Många operationer och några metoder för att bygga upp uttryck i olika typer,
  - if- och switch kommandon,
  - for- och while kommandon,
  - tilldelning och System.out.println,
  - fält
  - standard input
  - omdirigering av standard input och standard output
  - statiska metoder,
  - klasser som programbibliotek
  - att rita från ett Java program
- Inför tentan måste du kunna
  - att det finns både kommandon och uttryck
  - att det finns olika typer
  - förklara vad de kommandon vi har studerat gör
  - förklara vilka typer och värden några uttryck har
  - använda uttryck, tilldelning, System.out.println, if, switch, for och while i enkla program samt förklara vad program som är uppbyggda med dessa gör när de körs.
  - förklara hur man deklarerar, skapar, initierar och går igenom fält,
  - förklara varför programmet måste gå igenom fält för att skriva ut eller jämföra fältets element,
  - använda fält i enkla program och förklara hur programmet fungerar
  - använda standard input och omdirigering av standard input och standard output
  - vad en statisk metod i Java är och hur de kan användas för att definiera funktioner och metoder i Java
  - definiera och använda statiska metoder i ett program
  - förklara vad en klass bibliotek är samt definiera statiska metoder i en klass bibliotek
  - använda statiska metoder och klassbibliotek för att modularisera program

  Efter inlämningsuppgifterna finns några uppgifter från gamla tentor som handlar om detta: de finns bara för att du ska kunna bekanta dig med uppgifter av tenta typ, de ska INTE lämnas in! Har du frågor om de kan du ta upp det på Drop-in passet!

Inlämningsuppgifter - del 1

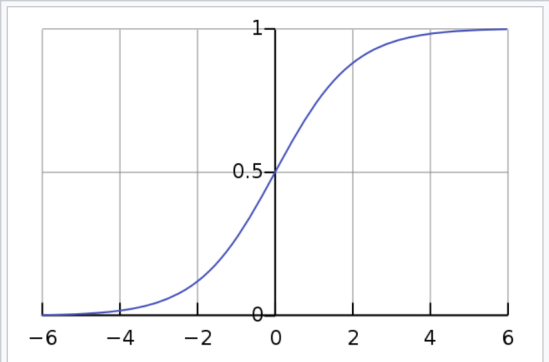
1. I filen *E1.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet ska användas för att beräkna värdet av en *logistisk* funktion. Du behöver inte förstå vad dessa matematiska funktioner kan användas till, men om du är intresserad finns det en utförlig artikel i Wikipedia (en.wikipedia.org/wiki/Logistic_function).
   För att kunna definiera funktionen behövs tre parametrar, benämnda med L, k, x_0:

   - L står för funktion värdenas supremum
   - k står för hur brant kurvan är
   - x_0 är x värdet för funktionens mittpunkt

   Givet dessa parametrar är funktionen

   $$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

   Figuren (från Wikipedias artikel) visar funktionen för parametrarna 1, 1, 0:

   Standard logistic function where $L = 1, k = 1, x_0 = 0$

   Programmet `E1` läser in parametrarna till en logistisk funktion från kommandoraden och beräknar värdet av funktionen i intervallet $[-1, 1]$ där x tar värdena $-1, -0.9, -0.8, \ldots, 0.9, 1$. Här ser du några exempel av hur programmet kan användas.
   Först med parametrar 1, 1, och 0:

```
java E1 1 1 0
0.2689414213699951
0.289050497374996
0.31002551887238755
0.33181222783183395
0.35434369377420455
0.3775406687981454
0.401312339887548
0.425557483188341
0.45016600268752216
0.47502081252106
0.5
0.52497918747894
0.549833997312478
0.574442516811659
```

```
0.598687660112452
0.6224593312018546
0.6456563062257954
0.668187772168166
0.6899744811276125
0.7109495026250039
0.7310585786300049
```

Nu med parametrar 2, 1 och 0:

```
java E1 2 1 0
0.5378828427399902
0.578100994749992
0.6200510377447751
0.6636244556636679
0.7086873875484091
0.7550813375962908
0.802624679775096
0.851114966376682
0.9003320053750443
0.95004162504212
1.0
1.04995837495788
1.099667994624956
1.148885033623318
1.197375320224904
1.2449186624037092
1.2913126124515908
1.336375544336332
1.379948962255225
1.4218990052500078
1.4621171572600098
```

I Java använder vi statiska metoder för att implementera funktioner. Vi använder metodens argument både för parametrarna och för x värdet.
Det du behöver göra är:

(a) Spara filen *E1.java* i mappen där du ska arbeta. Kompilera och kör programmet med tre kommandoradsargument. Du borde inte se någon utskrift.

(b) Komplettera koden för den statiska metoden `logistic` med signaturen

```
public static double logistic(double L, double k, double x0, double x)
```

Metoden ska implementera den logistiska funktionen $f(x)$ med parametrar `L`, `k`, `x0`.

(c) Komplettera metoden `main` så att den skriver ut värdena för funktionen med de parametrar som användaren anger i kommandoraden och för alla `x` värden i fältet `x_values`.

2. Filen *NumericalArrays.java* innehåller java kod som inte är tänkt att lösa något specifikt problem. Vi säger inte att denna kod är ett program, istället kallar vi det för ett bibliotek av metoder och funktioner (ett programbibliotek).

   Filens innehåll är alltså en Java klass, men den är inte tänkt att användas som ett självständigt program. Istället är den tänkt att samla ett antal funktioner och metoder som kan användas i flera program.

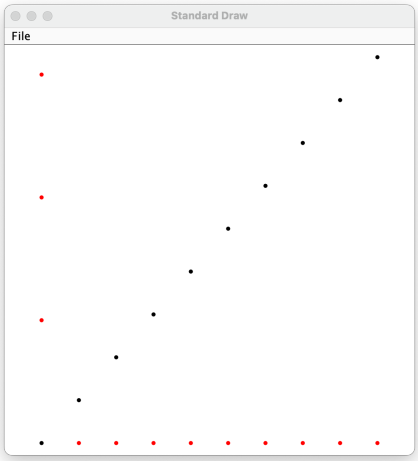
   I klassen `NumericalArrays` är alla funktioner och metoder hjälpmedel för program som behöver arbeta med fält av tal.

   När man programmerar en klass som är ett programbibliotek brukar man ha en `main` metod ändå. Syftet är att man ska kunna köra ett litet program som demonstrerar vad funktionerna och metoderna i klassen kan göra.

   När du har programmerat färdigt klassen `NumericalArrays` ska den kunna köras utan kommandoradsargument och resultera i följande utskrift. **Observera att `main` är redan programmerat, det ska du inte ändra i!**

```
java NumericalArrays

x is linspace(0, pi, 10):
0.0 0.3490658503988659 0.6981317007977318 1.0471975511965976 1.3962634015954636
1.7453292519943295 2.0943951023931953 2.443460952792061 2.792526803190927
3.141592653589793

a is repeat(10, 5):
5 5 5 5 5 5 5 5 5 5

a is now range(-10,10):
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10

min(x): 0.0  max(x): 3.141592653589793
min(a): -10  max(a): 10
x equal linspace(0,pi,10)? true
repeat(3,5) equal range(3,5)? false
sum(repeat(3,5)) = 15
```

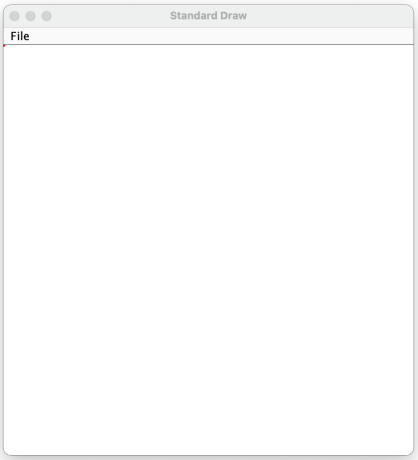
   Programmet öppnar också en rityta (`StdDraw`) med följande utseende:



Det du behöver göra är:

(a) Spara, kompilera och kör. Du bör se följande utskrift och rityta:

```
java NumericalArrays

x is linspace(0, pi, 10):

a is repeat(10, 5):

a is now range(-10,10):

min(x): 0.0  max(x): 0.0
min(a): 0  max(a): 0
x equal linspace(0,pi,10)? true
repeat(3,5) equal range(3,5)? true
sum(repeat(3,5)) = 0
```



(b) Komplettera definitionerna av de statiska metoderna som listas. Vad metoderna ska göra finns beskrivet i filen *NumericalArrays.java* med en kommentar direkt ovanför metoden.
Observera att det redan finns en funktion definierad (`linspace`) som kan användas för att skapa fält. `Linspace` förklarades i veckans föreläsningar.
De metoder du ska implementera är de 17 för vilka vi visar signaturen här. Förklaringar finns i en kommentar ovanför varje metod i filen *NumericalArray.java*.
Alla metoder är statiska. Fråga gärna om du inte förstår vad en metod ska göra.

```
public static void println(double[] numarray)

public static void println(int[] numarray)

public static int[] range(int a, int b)

public static int[] repeat(int n, int a)

public static double min(double[] numarray)

public static double max(double[] numarray)
```

```
public static int min(int[] numarray)

public static int max(int[] numarray)

public static double sum(double[] numarray)

public static int sum(int[] numarray)

public static double average(double[] numarray)

public static double standardDeviation(double[] numarray)

public static double dotproduct(double[] a, double[] b)

public static boolean equal(double[] a, double[] b)

public static boolean equal(int[] a, int[] b)

public static void plot(double[] numarray)

public static void plot(int[] numarray)
```

Inlämningsuppgifter - del 2

I filen *NumericalArrays.java* finns ytterligare en metod som ska implementeras. Det är denna metod som är bonus uppgiften.

```
public static double[]  runningAverage(double[] values, int window)
```

Metoden ska implementera en så kallade *running* (eller *moving*) *average*. Definitionen (från Wikipedias artikel en.wikipedia.org/wiki/Moving_average) är:

> Given a series of numbers and a fixed subset size, the first element of the moving average is obtained by taking the average of the initial fixed subset of the number series. Then the subset is modified by "shifting forward"; that is, excluding the first number of the series and including the next value in the subset.

We call the fixed subset size a window.

Uppgifter från gamla tentor som kan lösas efter vecka 4.

1. Här ser du signaturen för en metod som räknar ut summan av alla element i ett fält:

   ```
   public static int sum(int[] a)
   ```

   (a) Ange metodens namn, resultattyp och argumenttyp.

   Metodens namn är `sum`, argumenttypen är `int[]` och resultattypen är `int`.

   (b) Skriv ett kodfragment som skapar ett fält med elementen 1,2,3,4 och 5 och beräknar summan av dessa element med hjälp av metoden `sum`.
   OBS! Du behöver inte programmera `sum`: du vet att den är redan programmerad. Du ska bara visa att du kan använda den.

   Följande kodfragmentet skapar fältet i samband med en variabeldeklaration och skriver ut värdet av summan av fältets element. Summan beräknas med ett anrop till metoden `sum`. Vi antar att kodfragmentet används i samma `class` där metoden är definierad i.

   ```
   int[] numbers = {1,2,3,4,5};
   System.out.println(sum(numbers));
   ```

2. Programmera en metod med signatur

   ```
   public static int[] keep(int x, int []a)
   ```

   som beräknar ett fält med bara de första `x` elementen i fältet `a`.

   Till exempel ska `keep(3, [9,8,7,6,5,4,3,2,1])` resultera i fältet `[9,8,7]`.

   I min implementation av `keep` gör jag antagandet att `x <= a.length`: metoden kan användas för att behålla från 0 till alla element i fältet.

   ```
   public static int[] keep(int x, int[] a){
      int[] res = new int[x];
      for(int i = 0; i < x; i++){
          res[i] = a[i];
      }
      return res;
   }
   ```

   Resultatfältet har som längd antalet element från argumentfältet som ska behållas. En `for`-loop används för att kopiera till resultatet de första elementen från argumentfältet.

3. Programmera en metod med signatur

   ```
   public static boolean isConstant(int[] a)
   ```

   som kan användas för att testa om alla element i fältet `a` är lika.

   Till exempel ska `isConstant([2,2,2])` resultera i `true` medan `isConstant([1,2,2])` ska resultera i `false`.

   Implementationen:

```
public static boolean isConstant(int[] a){
   for(int i = 0; i < a.length-1; i++){
       if(a[i] != a[i+1]){
           return false;
       }
   }
   return true;
}
```

Implementationen jämför varje element med nästa element (därför avslutas `for`-loopen vid näst sista element). Så fort ett element som inte är likt föregående upptäcks avslutas metoden med resultatet `false`. Om alla element är lika (`for`-loopen körs genom hela fältet) avslutas metoden med resultatet `true`.

4. Här ser du signaturen för en metod som byter plats på de element som finns i positionerna i och j i ett fält:

   ```
   public static void exchange(int[] a, int i, int j)
   ```

   (a) Ange metodens namn, resultattyp och argumenttyp. Observera att det finns flera argument, du ska ange alla argumenttyperna.

   Lösningsförslag:
   Metodens namn är `exchange`, resultattypen är `void` och argumenttyperna är `int[]`, `int` och `int`.

   (b) Skriv ett kodfragment som skapar ett fält med elementen 10, 20, 50, 40 och 30 och använder metoden för att byta plats på 50 och 30. **OBS!** Du behöver inte programmera `exchange`: du vet att den är redan programmerad. Du ska bara visa att du kan använda den.

   Lösningsförslag:

   ```
   int [] a = {10, 20, 50, 40, 30};
   exchange(a, 2, 4);
   ```

   Efter att ha skapat fältet anropas metoden med fältet och positionerna för de element som ska byta plats.

5. Programmera en metod med signatur

   ```
   public static int max(int []a)
   ```

   som beräknar det största talet i fältet `a`.

   Till exempel ska `max([8,7,6,9,5,4,3,2,1])` resultera i värdet `9`.

   Lösningsförslag:

   ```
   public static int max(int[] a){
      int m = a[0];
      for(int i = 1; i < a.length; i++){
         if(a[i] > m){
            m = a[i];
         }
   ```

```
    }
    return m;
}
```

Variablen `m` initieras till första värdet i faltet för att sedan uppdateras så att den hela tiden är den största värdet som har setts i fältet upp till den plats som for-loopen har hunnit till. När for-loopen är klar, innehåller `m` största värdet i fältet och det blir metodens resultat.

6. Programmera en metod med signatur

   ```
   public static void println(int[] a)
   ```

   som skriver ut fältets längd på en rad och sedan alla element i fältet på nästa rad.

   Till exempel ska `println([2,2,2])` skriva ut

   ```
   3
   2 2 2
   ```

   Lösningsförslag:

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