Föreläsning 3-4

Innehåll

- Skriva egna metoder
- Logiska uttryck
- Vektorer
- Algoritm för att beräkna min och max
- Algoritm för sökning

Diskutera

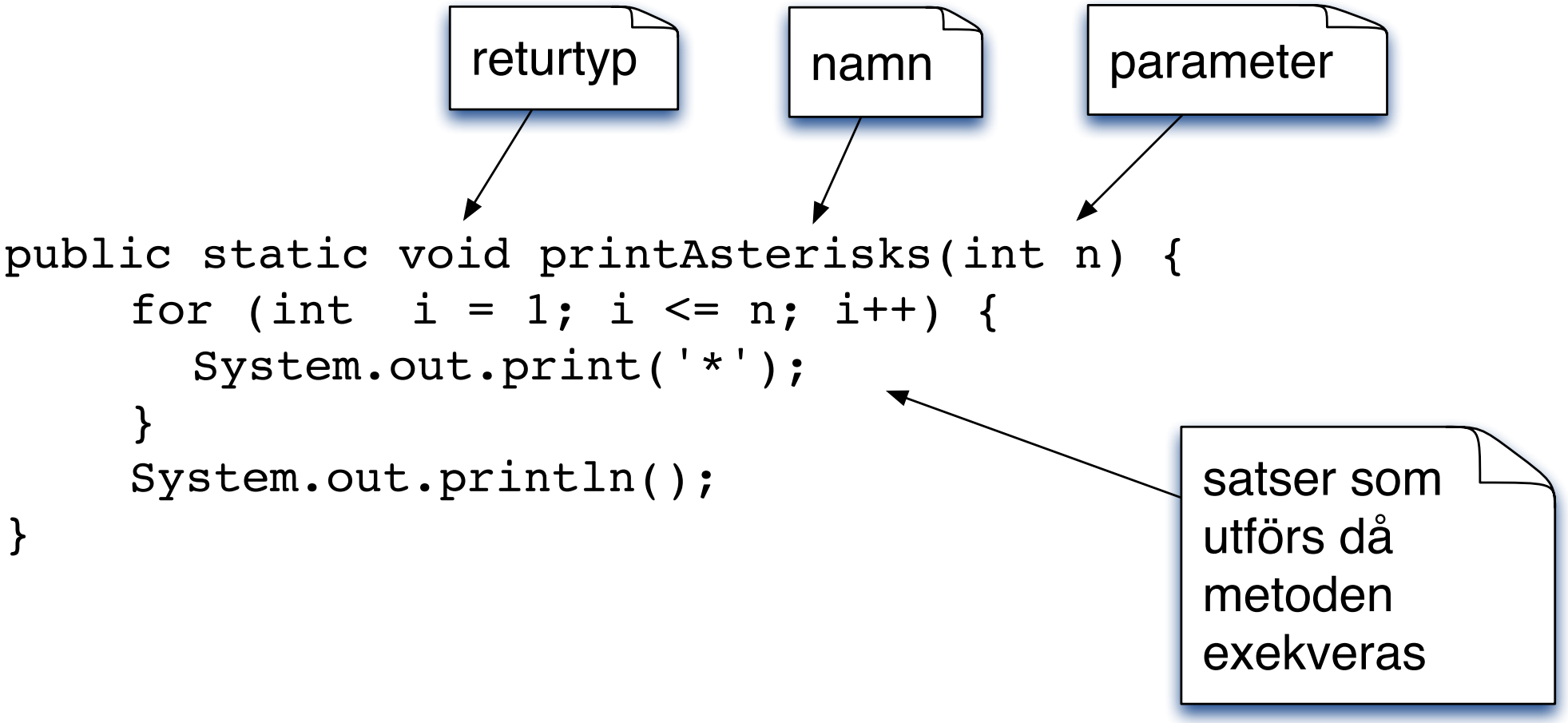
- Vad gör programmet programmet?
- Föreslå vilka satser vi kan bryta ut till en egen metod.

```
public class Asterisks {
    public static void main(String[] args) {
        System.out.println("Antal: ");
        Scanner scan = new Scanner(System.in);
        int nbr = scan.nextInt();
        for (int i = 1; i <= nbr; i++) {
            System.out.print('*');
        }
        System.out.println();
    }
}
```

Programexempel med metod

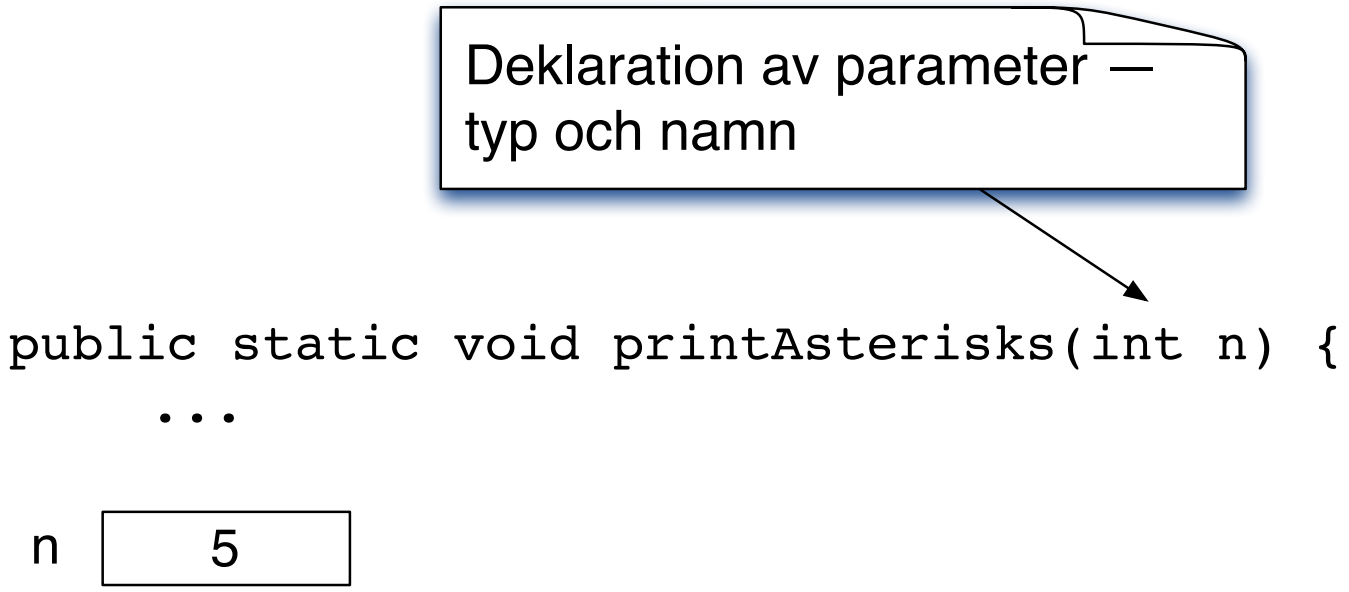
```
public class Asterisks {
    public static void main(String[] args) {
        System.out.println("Antal: ");
        Scanner scan = new Scanner(System.in);
        int nbr = scan.nextInt();
        Asterisks.printAsterisks(nbr); // eller bara
                                       // printAsterisks();
                                       // om metoden finns i samma klass
    }

    public static void printAsterisks(int n) {
        for (int  i = 1; i <= n; i++) {
            System.out.print('*');
        }
        System.out.println();
    }
```

Metod


returtyp
namn
parameter
public static void printAsterisks(int n) {
for (int i = 1; i <= n; i++) {
System.out.print('*');
}
System.out.println();
}
satser som utförs då metoden exekveras

Parametrar

- Parametrarna deklareras i metodens parameterlista.
- Parametrarna är bara åtkomliga inuti metoden.
  - Parametrarna skapas då metoden anropas och försvinner när metoden är färdigexekverad.
- Argumenten kopieras vid anropet in i motsvarande parametrar.



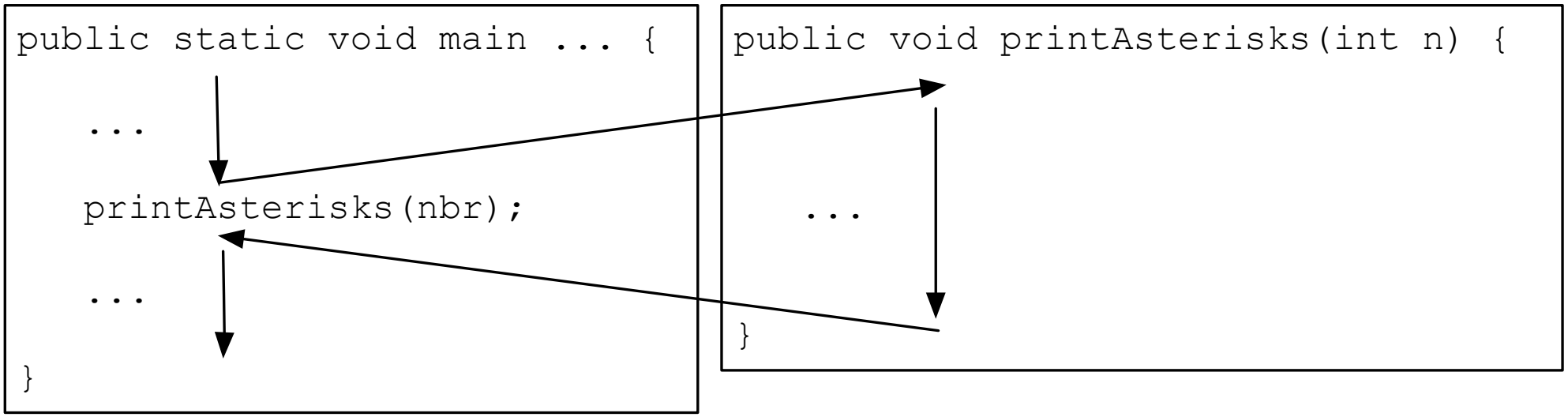
Modifierare för metoder

- `public` – Metoden kan även användas utanför klassen.
- `private` – Metoden kan bara användas inuti klassen.

- `static` – Metoden hör till klassen och anropas inte på något objekt.

Exekvering och metodanrop

- Ett anrop av en metod innebär att exekveringen fortsätter med den första satsen i den anropade metoden.
- När den anropade metoden är klar återupptas exekveringen i den metod där anropet gjordes.



Diskutera

Hur ska man ändra programmet nedan så att det skriver ut en godtycklig text ett önskat antal gånger?

```
public class Asterisks {
    public static void main(String[] args) {
        System.out.println("Antal: ");
        Scanner scan = new Scanner(System.in);
        int nbr = scan.nextInt();
        Asterisks.printAsterisks(nbr);
    }

    public static void printAsterisks(int n) {
        for (int i = 1; i <= n; i++) {
            System.out.print('*');
        }
        System.out.println();
    }
```

Returtyp

- `void` betyder att metoden inte returnerar något värde.
- Om metoden returnerar ett värde ska den ha en datayp (`int`, `double`, `String` ...,) som returtyp.
  - Metoden måste då också ha (minst) en `return`-sats. Exempel:

```
private static double calcArea(double radius) {
        return Math.PI * radius * radius;
}
```

Metoder som returnerar värden

Programexempel

```
public class CircleProgram {
    public static void main(String[] args) {
        System.out.println("Radie: ");
        Scanner scan = new Scanner(System.in);
        double radius = scan.nextDouble();
        System.out.println("Area: " + calcArea(radius));
        System.out.println("Omkrets: " + calcCircumference(radius));
    }

    public static double calcArea(double radius) {
        return Math.PI * radius * radius;
    }

    public static double calcCircumference(double radius) {
        return 2 * Math.PI * radius;
    }
}
```

Konstanter

- I klassen `Math` finns **konstanten** `Math.PI`:

```
public static final double PI = 3.14159265358979323846;
```

- Man kan deklarera egna konstanter för värden som inte ska ändras. Exempel:

```
static final int NBR_CARDS = 52;
```

- `final` betyder att värdet inte får ändras.

Block

- Satser som hör ihop.
- Omges av klammerparenteser { }

Exempel:

```
public class Asterisks {
    ...   // Satserna i en klass
}
```

```
public static void printAsterisks(int n) {
    ...  // Satserna i en metod
}
```

```
for (int i = 1; i <= n; i++) {
    ...  // Satser som utföras i en for-sats
}
```

Kommentar

- Metoder ska egentligen ha dokumentationskommentarer som beskriver:
  - vad metoden gör.
  - vad parametrarna används till.

Exempel:

```
/** Skriver ut en rad med n st asterisker. */
...
```

eller mer noggrant (med javadoc-taggar)

```
/**
 * Skriver ut en rad med n st asterisker.
 *
 * @param n
 *             antal asterisker som ska skrivas ut
 */
...
```

Övning

Skriv klart metoden `isEven`. Ledtråd: använd operatorn %.

```
public class EvenNumbers {
    public static void main(String[] args) {
        System.out.println("Skriv ett heltal: ");
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        if (isEven(n)) {
            System.out.println(n + " är ett jämnt tal.");
        } else {
            System.out.println(n + " är ett udda tal.");
        }
    }

    private static boolean isEven(int n) {



    }
}
```

Logiska uttryck

- Logiska uttryck används bland annat som villkor i if- och while-satser:

```
if (x < 5)  {
    ...
}

while (true) {
    ...
}
```

- Logiska uttryck kan ha två möjliga värden, `true` eller `false`.
- Variabler av typen `boolean` kan tilldelas logiska uttryck:

```
boolean ok = true;
ok = false;
ok = x < 5;
```

Relationer

- Ett logiskt uttryck kan vara en relation där man jämför värdet av två uttryck: `x < 5`
- Relationsoperatorer:

  `<`

  `<=`

  `==` "lika med"

  `>=`

  `>`

  `!=` "skilt från"

Logiska operatorer

- Logiska uttryck kan kopplas samman med operatorerna:

  `!` "icke"
  `&&` "och"
  `||` "eller"

- Exempel:

  x tillhör intervallet [5, 10] `x >= 5 && x <= 10`
  x tillhör *inte* intervallet [5, 10] `x < 5 || x > 10`

- Ett och-uttryck är `true` om *alla* deluttryck har värdet `true`.
- Ett eller-uttryck är `true` om *minst ett* av deluttrycken har värdet `true`.

de Morgans lagar

- Antag att A och B är två logiska uttryck. Då gäller
  `! (A && B)` motsvarar `! A || ! B`
  `! (A || B)` motsvarar `! A && ! B`

- Att x tillhör intervallet [5, 10] kan skrivas

```
x >= 5 && x <= 10
```

  Att x *inte* tillhör intervallet [5, 10] är negationen till ovanstående:

```
! (x >= 5 && x <= 10)
```

  Men det kan förenklas till:

```
x < 5 || x > 10
```

Diskutera

Betrakta följande program:

```
public class Minimum {
    public static void main(String[] args) {
        System.out.println("Skriv två heltal: ");
        Scanner scan = new Scanner(System.in);
        int n1 = scan.nextInt();
        int n2 = scan.nextInt();
        int min;
        if (n1 < n2) {
            min =  n1;
        } else {
            min =  n2;
        }
        System.out.println("Minsta talet är " + min);
    }
}
```

Vad händer om talen är lika?

Hur ska vi göra om indata består av ett godtyckligt antal tal?

Sök minsta talet

Mönster

- Uppgift: Beräkna det minsta talet i en följd av tal.
- Lösning: Deklarera en variabel `min` som ska hålla reda på det hittills minsta värdet. Låt `min` få ett stort startvärde. Gå sedan igenom talen och jämför med `min`. Uppdatera `min` ifall det aktuella talet är mindre.
- Algoritmen i pseudokod:

```
min = "stort värde";
för alla värden {
    value = "nästa värde";
    if (value < min) {
        min = value;
    }
}
```

Sök minsta talet

Exempel

```
public class ComputeMin {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int min = Integer.MAX_VALUE;
        while (scan.hasNextInt()) {
            int nbr = scan.nextInt();
            if (nbr < min) {
                min = nbr;
            }
        }
        System.out.println("Minsta talet är " + min);
    }
}
```

Konstanter för minsta och största tal

- Konstanter för minsta respektive största värde:

```
int i1 = Integer.MIN_VALUE;     // -2147483648
int i2 = Integer.MAX_VALUE;     // 2147483647

double d1 = Double.MIN_VALUE;   // 4.9E-324 (OBS! minsta
                                   möjliga positiva flyttal)
double d2 = Double.MAX_VALUE;   //  1.8E308
double d3 = -Double.MAX_VALUE   // -1.8E308
```

Med `E` menas ”gånger 10 upphöjt till”

Beräkna maximum

Mönster

- Uppgift: Beräkna det största talet i en följd av tal.
- Lösning: Deklarera en variabel `max` som ska hålla reda på det hittills största värdet. Låt `max` få ett litet startvärde. Gå sedan igenom talen och jämför med `max`. Uppdatera `max` ifall det aktuella talet är större.
- Algoritmen i pseudokod:

```
max = "litet värde";
för alla värden {
    value = "nästa värde";
    if (value > max) {
        max = value;
    }
}
```

Beräkna maximum

Övning

Beräkna och skriv ut det största talet av ett antal heltal som läses in från tangentbordet. Fyll i den kod som saknas.

```
public class ComputeMax {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int max = Integer.MIN_VALUE;
        while (scan.hasNextInt()) {
            int nbr = scan.nextInt();









        }
        System.out.println("Största talet är " + max);
    }
}
```

Diskutera

```
public class ArrayExample {
    public static void main(String[] args) {
        int[] v = {10, 25, -10, 42, 67, -23, 100, 5, 0};
        int min = Integer.MAX_VALUE;
        for (int i = 0; i < v.length; i++) {
            if (v[i] < min) {
                min = v[i];
            }
        }
        System.out.println("Det minsta talet är " + min);
    }
}
```

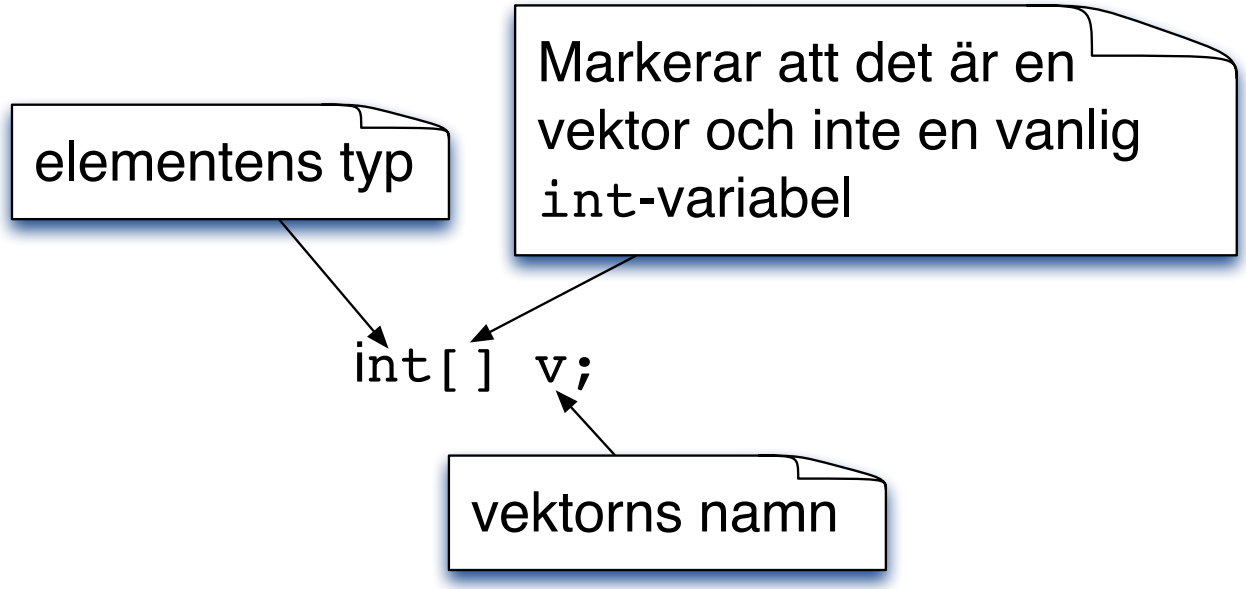
Vad gör programmet?

Deklarera vektorer

- Vektorer deklareras precis som andra variabler med

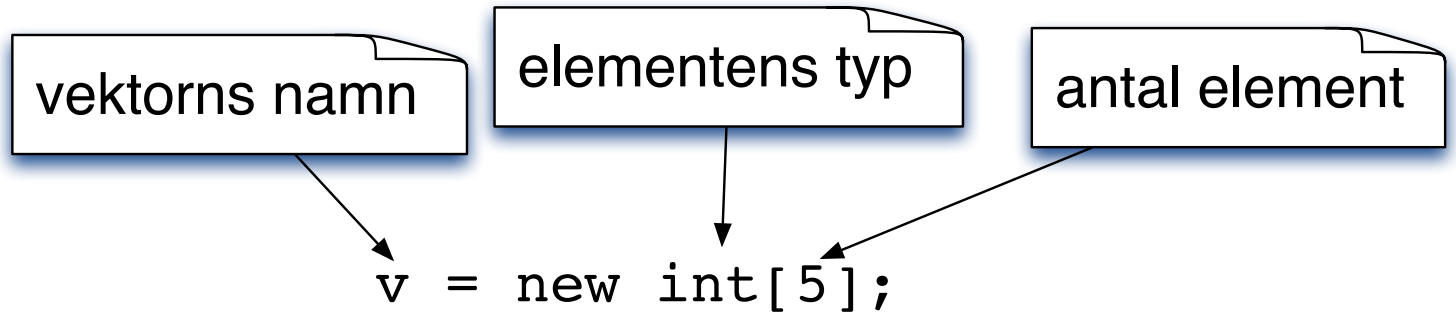
```
datatyp namn
```

- Exempel:



v

Skapa vektorer

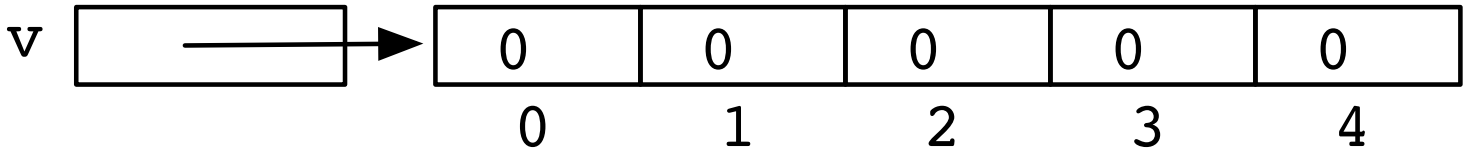


- Antal element måste anges.
- Elementen får automatiskt startvärden.
- Oftast deklarerar och skapar man vektorn på samma gång:

```
int[] v = new int[5];
```

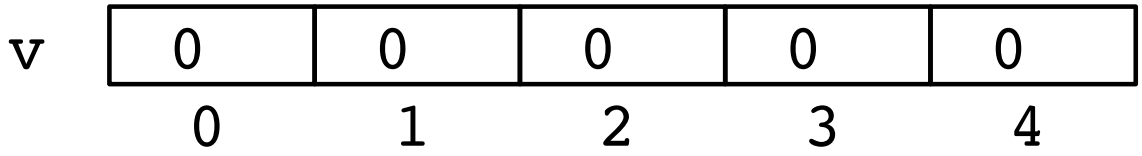
Vektorer är objekt

- Vektorer är objekt i Java. `v` refererar till vektorn.

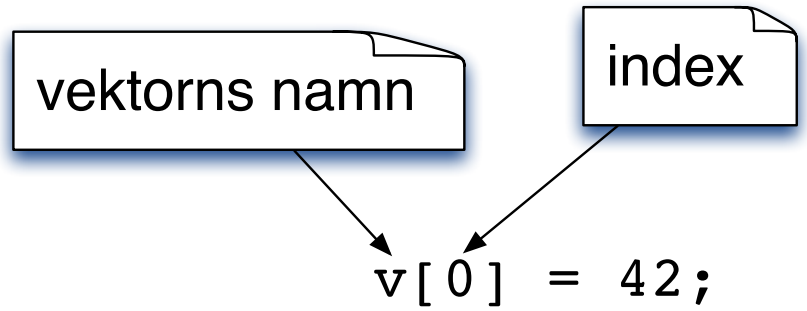


  Man säger att variabeln `v` refererar till vektorn.

- Men ofta ritar man en enklare bild av vektorn:



Använda vektorelement



- Ett vektorelement används som en vanlig variabel.
- Det är viktigt att indexet håller sig inom gränserna $[0, \mathit{antal\ element} - 1]$

Ta reda på vektorns längd

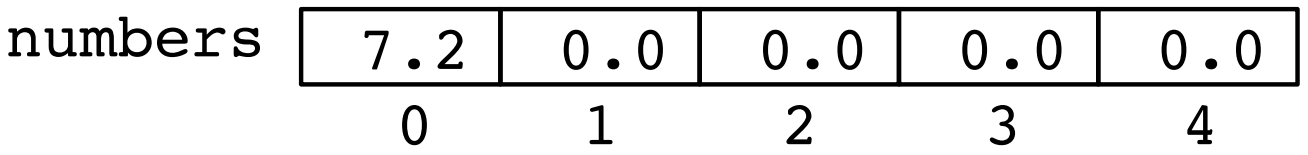
- Använd `length` för att ta reda antal element i vektorn.

```
for (int i = 0; i < v.length; i++) {
    v[i] = scan.nextInt();
}
```

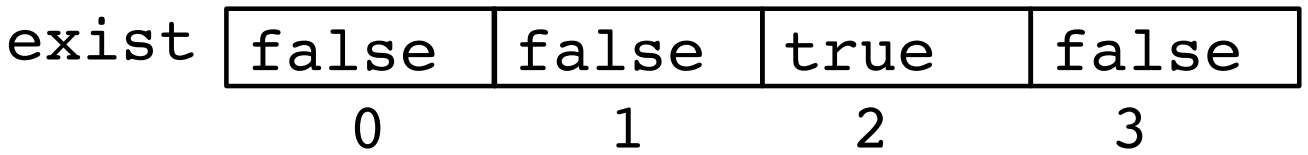
- Tänk på att elementen numreras från 0 och uppåt.

Övning

Deklarera och skapa en vektor `numbers` med plats för 5 stycken element av typen `double`. Tilldela första elementet värdet 7.2.



Deklarera och skapa en vektor `exist` med plats för 4 stycken element av typen `boolean`. Tilldela tredje elementet värdet `true`.



Genvägar

Överkurs, men praktiskt

- Skapa en vektor med ett givet innehåll:

```
int[] a = {7, 8, 9};
```

- Metoden `Arrays.toString` returnerar en sträng med vektorns innehåll. Skriv ut innehållet i vektorn:

```
System.out.println(Arrays.toString(b));
```

Linjärsökning

Mönster

- *Uppgift*: Sök upp givet element i en följd av element.
- *Lösning:* Gå igenom elementen i tur och ordning och kontrollera för varje element om det är det sökta. Avbryt om det sökta elementet påträffas.
- Algoritmen i pseudokod:

```
for (int i = 0; i < "antal element"; i++) {
    if ("elementet på plats i är det vi söker") {
        avbryt
    }
}
```

Linjärsökning

Exempel

```
/** Söker efter talet nbr i vektorn v. Om nbr finns
    returneras platsen för nbr, annars -1 */
public static int indexOf(int[] v, int nbr) {
    for (int i = 0; i < v.length; i++) {
        if (v[i] == nbr) {
            return i;
        }
    }
    return -1;
}
```

Diskutera

```
public static int indexOf(int[] v, int nbr) {
   for (int i = 0; i < v.length; i++) {
      if (v[i] == nbr) {
         return i;
      } else {
         return -1;
      }
   }
}
```

Varför fungerar inte det här?

Checklista

Exempel på vad du ska kunna

- Skriva egna metoder.
- Formulera logiska uttryck i Java.
- Deklarera, skapa och använda vektorer av typen `int[]`, `double[]`, `boolean[]` ...
- Använda en `for`-sats för att traversera en vektor (dvs. ”gå igenom ” vektorn och göra något med elementen).
- Formulera algoritmer och programkod för att beräkna minsta och största värde.
- Formulera algoritmer och programkod för sökning.