EDAA20
Programmering och databaser

Föreläsning 4, **Vektorer**, 2023-09-05, Niklas Fors

Många värden

Lagra temperatur för 7 dagar med 7 variabler:

```
double tempDay1 = 0.8;
double tempDay2 = 0.3;
...
double tempDay7 = -2.1;
```

Istället med ***vektor***:

```
double[] temp = {0.8, 0.3, 0.9, 2.4, 2.0, 0.8, -2.1};
```

Vektor

- En ***vektor*** (*array*) är en följd av element av en viss typ och med en storlek
- Exempel:

```
int[] v = new int[5];  // int-vektor med 5 element
```

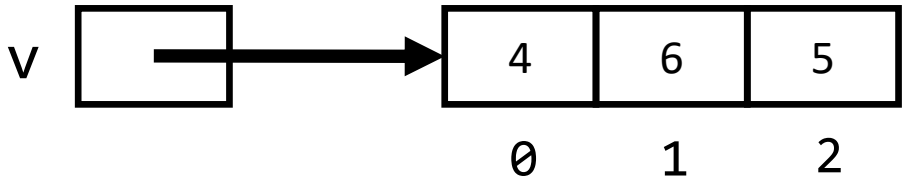
- Elementen får automatiskt startvärden (`0/0.0/false`)

- Skapa vektor med givna startvärden:

```
int[] v = {4, 6, 5};   // int-vektor med 3 element
```

Vektor – illustration

```
int[] v = {4, 6, 5};
```



Första elementet är alltid på **plats 0**!

Man säger att variabeln **v** *refererar* till vektorn.

Element

Tilldela element ett värde:

```
v[0] = 42;
```

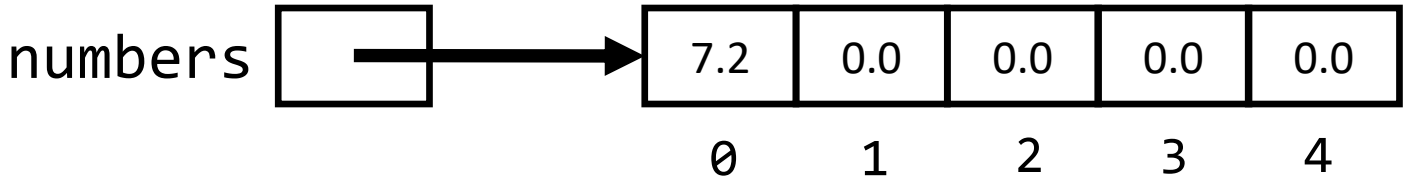
Elementets ***index*** är inom intervallet 0 .. *storlek*-1

Hämta element**:**

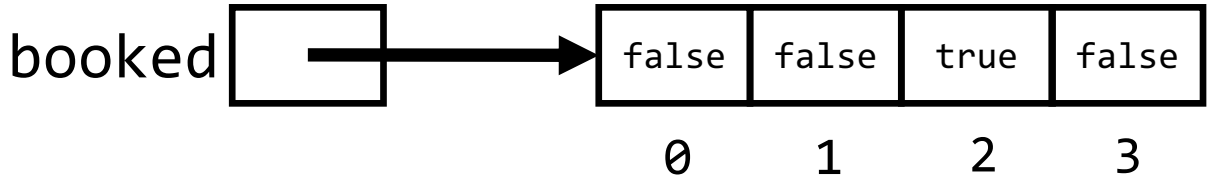
```
if (v[2] > 5) {
  x = v[0] / 2;
}
```

Övning

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

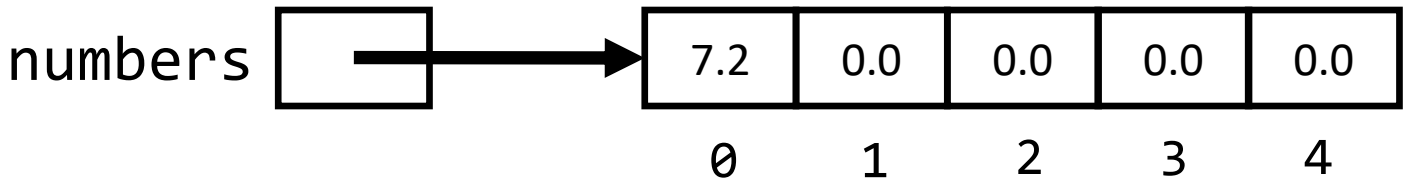


Deklarera och skapa en vektor **`booked`** med plats för 4 element av typen **`boolean`**. Tilldela tredje elementet värdet `true`.



Lösningsförslag 1/2

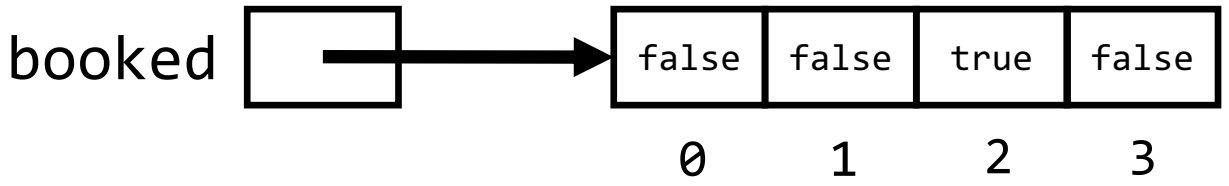
Deklarera och skapa en vektor **`numbers`** med plats för 5 element av typen **`double`**. Tilldela första elementet värdet 7.2.




```
double[] numbers = new double[5];
numbers[0] = 7.2;
```

Lösningsförslag 2/2

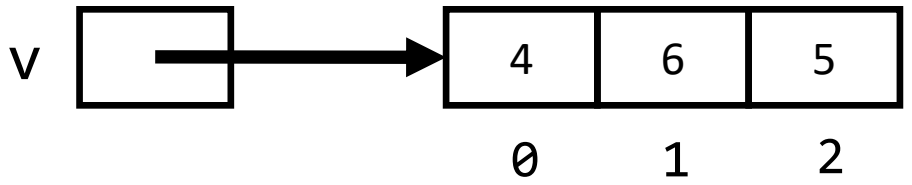
Deklarera och skapa en vektor **booked** med plats för 4 element av typen **boolean**. Tilldela tredje elementet värdet `true`.




```
boolean[] booked = new boolean[4];
booked[2] = true;
```

Summera element, försök 1

```
int[] v = {4, 6, 5};
```

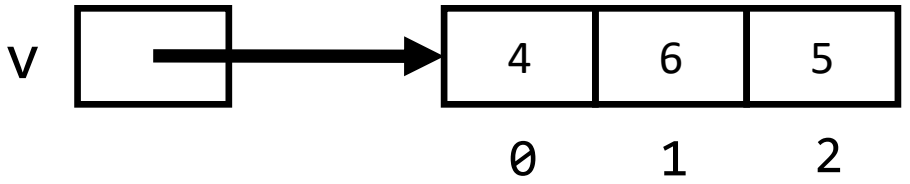



```
int sum = 0;

sum = v[0] + v[1] + v[2];

System.out.println(sum);  // 15
```

Koden ej generell!
Hanterar enbart en vektor med 3 element

Summera element, försök 2

```
int[] v = {4, 6, 5};
```



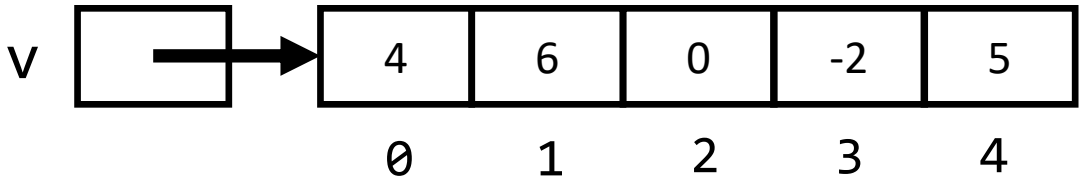

```
int sum = 0;
for (int i = 0; i < v.length; i++) {
   sum = sum + v[i];
}
System.out.println(sum);  // 15
```

Vektorns storlek: **`v.length`**

Notera att i=0 och **<** används i `for`-satsen (pga index börjar på 0)!

Övning

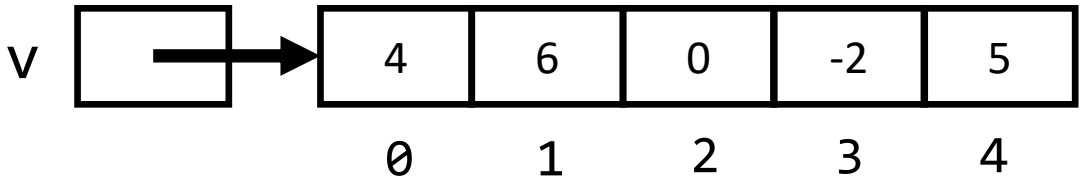
```
int[] v = {4, 6, 0, -2, 5};
```




```
// Ändra på koden nedan för att räkna
// antalet positiva tal i vektorn v.
int sum = 0;
for (int i = 0; i < v.length; i++) {
   sum = sum + v[i];
}
System.out.println(sum);
```

Övning – lösning

```
int[] v = {4, 6, 0, -2, 5};
```




```
// Ändra på koden nedan för att räkna
// antalet positiva tal i vektorn v.
int n = 0;
for (int i = 0; i < v.length; i++) {
  if (v[i] > 0) {
    n++;
  }
}
System.out.println(n);   // 3
```

Hitta största tal

Algoritm: hitta största tal

Indata: en sekvens av tal
Utdata: största talet, *max*
max = *<ett väldigt litet tal>*
för varje tal *x* i sekvensen
    om *x* > *max*
        max = *x*

Hitta största tal

Algoritm: hitta största tal

Indata: en sekvens av tal
Utdata: största talet, *max*
max = *<ett väldigt litet tal>*
för varje tal *x* i sekvensen
    om *x* > *max*
        max = *x*

Skillnader jämfört med algoritmen för att hitta minsta talet
(och variabelnamnet max)

Största talet i en `int`-vektor

```
public class ArrayMax {
  public static void main(String[] args) {
    int[] a = { 10, 25, -10, 42, 67, -23, 100, 5, 0 };
    int max = Integer.MIN_VALUE;
    for (int i = 0; i < a.length; i++) {
      if (a[i] > max) {
        max = a[i];
      }
    }
    System.out.println("Största tal: " + max);
  }
}
```

Bryt ut till metod

```
public class ArrayMax {
  public static void main(String[] args) {
    int[] a = { 10, 25, -10, 42, 67, -23, 100, 5, 0 };
    System.out.println("Största tal: " + maxValue(a));
  }

  public static int maxValue(int[] v) {
    int max = Integer.MIN_VALUE;
    for (int i = 0; i < v.length; i++) {
      if (v[i] > max) {
        max = v[i];
      }
    }
    return max;
  }
}
```

Övning, dubblera heltal

```
public class DoubleElements {
  public static void main(String[] args) {
    int[] a = {4, 6, 5, 2};
    doubleElements(a);

    for (int i = 0; i < a.length; i++) {  // Skriv ut alla element
       System.out.println(a[i]);
    }
  }

  /** ÖVNING: Implementera följande metod som tar en vektor av heltal
              och dubblerar alla element i vektorn. */
  public static void doubleElements(int[] v) {



  }
}
```

Övning, dubblera heltal – lösning

```
public class DoubleElements {
  public static void main(String[] args) {
    int[] a = {4, 6, 5, 2};
    doubleElements(a);

    for (int i = 0; i < a.length; i++) {  // Skriv ut alla element
       System.out.println(a[i]);
    }
  }

  /** ÖVNING: Implementera följande metod som tar en vektor av heltal
              och dubblerar alla element i vektorn. */
  public static void doubleElements(int[] v) {
    for (int i = 0; i < v.length; i++) {
      v[i] = v[i] * 2;
    }
  }
}
```

Övning, dubblera heltal – lösning

```
public class DoubleElements {
  public static void main(String[] args) {
    int[] a = {4, 6, 5, 2};
    doubleElements(a);

    for (int i = 0; i < a.length; i++) {  // Skriv ut alla element
       System.out.println(a[i]);
    }
  }



  public static void doubleElements(int[] v) {
    for (int i = 0; i < v.length; i++) {
      v[i] = v[i] * 2;
    }
  }
}
```

Referensen till vektorn **a** skickas med som argument

Element i vektorn ändras eftersom parametern **v refererar** till **samma** vektor som i `main`-metoden!

Linjärsökning

Problem: kontrollera om ett element finns i en sekvens av element

Indata: sekvens av element, sökt element
Utdata: index för sökt element eller -1 om ej finns
Pseudokod:

```
för varje element x i sekvensen
    om x är det sökta elementet
        avbryt sökningen och returnera index för x
returnera -1
```

Linjärsökning

Problem: kontrollera om ett element finns i en sekvens av element

Indata: sekvens av element, sökt element
Utdata: index för sökt element eller -1 om ej finns
Pseudokod:

```
för varje element x i sekvensen
    om x är det sökta elementet
        avbryt sökningen och returnera index för x
returnera -1
```

För att veta att det sökta elementet ***inte*** finns måste ***alla*** element gås igenom!
Dvs, man kan bara returnera -1 ***efter*** loopen

Linjärsökning – implementation

```
public class ArraySearch {
  public static void main(String[] args) {
    int[] a = { 10, 25, -10, 42, 67, -23, 100, 5, 0 };
    int index = indexOf(a, 67);
    System.out.println("67 finns på index: " + index);
  }

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

Övning

```
public class ArraySearch {
  public static void main(String[] args) {
    int[] a = { 10, 25, -10, 42, 67, -23, 100, 5, 0 };
    int index = indexOf(a, 67);
    System.out.println("67 finns på index: " + index);
  }

  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;
      }
    }
    return -1;
  }
}
```

Metoden `indexOf` innehåller nu ett vanligt fel.
Vad händer när programmet exekverar och varför?

FEL! För tidig return!

```
public class ArraySearch {
  public static void main(String[] args) {
    int[] a = { 10, 25, -10, 42, 67, -23, 100, 5, 0 };
    int index = indexOf(a, 67);
    System.out.println("67 finns på index: " + index);
  }

  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;
      }
    }
    return -1;
  }
}
```

Metoden returnerar för tidigt!

Metoden kommer *alltid* att returnera ett värde första varvet i `for`-satsen. Dvs, enbart första elementet kontrolleras och sedan avbryts sökningen!

För tidig `return` är ett vanligt fel i kursen och ger signifikant poängavdrag på tentan!

Inläsning från användaren

```
public class ArraySearch {
  public static void main(String[] args) {
    // Läs in 5 heltal från användaren och lägg in dem i vektorn:
    Scanner scan = new Scanner(System.in);
    int[] a = new int[5];
    for (int i = 0; i < a.length; i++) {
      a[i] = scan.nextInt();
    }

    int index = indexOf(a, 67);
    System.out.println("67 finns på index: " + index);
  }

  public static int indexOf(int v[], int nbr) {
     ... // som tidigare
  }
}
```

Linjärsökning – alternativ med **`break`**-sats

```
public class ArraySearch {
  public static void main(String[] args) {
    ...
  }

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

Satsen `break` avbryter närmaste `for`/`while`-sats.
Exekveringen fortsätter efter `for`-satsen

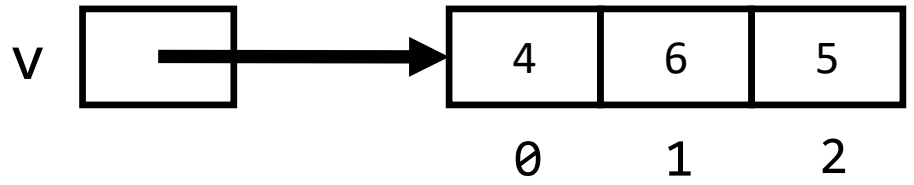
`return` vs `break`

En `return`-sats avbryter nuvarande metod och kan returnera ett värde

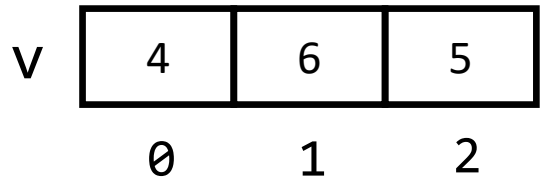
En `break`-sats avbryter närmaste `for`/`while`-sats

Förenkling när man ritar

Ibland när man ritar förenklar man följande figur:



Till:



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