Föreläsning 9-10

Innehåll

- Inläsning från textfil, utskrift från textfil
- Vektorer med objekt
- Matriser

Klassen Scanner

Läsa från `System.in`

- Vi har tidigare skapat `Scanner`-objekt som läser data som användaren skriver när programmet exekveras:
- Exempel på ett program som läser heltal och summerar dem.

```
public class SumNumbers {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in));
        int sum = 0;
        while (scan.hasNextInt()) {
            sum = sum + scan.nextInt();
        }
        System.out.println("Summa: " + sum);
    }
}
```

Klassen Scanner

Läsa en sträng eller från en fil

- Det finns flera konstuktorer i klassen `Scanner`. Man kan t.ex. skapa ett `Scanner`-objekt som ”läser” från en sträng:

```
String s = "1 2 3 4 5";
Scanner scan = new Scanner(s);
```

- Man kan också skapa `Scanner`-objekt som läser från en fil (här från en fil med namnet `numbers.txt`):

```
Scanner scan = new Scanner(new File("numbers.txt"));
```

  - `File` är en färdig klass i Java som representerar en fil.

try catch-sats

- När ett program exekveras som ska öppna en fil för läsning kanske det inte finns någon sådan fil.
- Detta fel (och andra liknande fel som är utanför programmerarens kontroll) måste man ta hand om i Java.
- Man kan använda en `try catch`-sats för detta.
- I exemplet nedan görs ett försök att öppna filen `numbers.txt`. Om det inte går avbryts exekveringen av satserna i `try`-blocket och satserna i `catch`-blocket kommer att utföras.

```
try {
    // Här försöker vi öppna filen numbers.txt
    scan = new Scanner(new File("numbers.txt"));
} catch (FileNotFoundException e) {
    // Här kan man skriva vad som ska hända om
    // filen inte kan öppnas.
}
```

Läsa indata från fil

Exempel

```
public class SumNumbers {
    public static void main(String[] args) {
        Scanner scan = null;
        try {
            scan = new Scanner(new File("numbers.txt"));
        } catch (FileNotFoundException e) {
            System.out.println("Filen kunde inte öppnas");
            System.exit(1);
        }
        int sum = 0;
        while (scan.hasNextInt()) {
            sum = sum + scan.nextInt();
        }
        System.out.println("Summa: " + sum);
        scan.close();   // stäng filen
    }
}
```

Utskrift på fil

Klassen `PrintWriter`

```
public class PrintWriterExample {
    public static void main(String[] args) {
        PrintWriter out = null;
        try {
            out = new PrintWriter(new File("utdata.txt"));
        } catch (FileNotFoundException e) {
            System.out.println("Filen kunde inte öppnas");
            System.exit(1);
        }
        // utskrifter med out.print hamnar på filen utdata.txt
        out.close();
    }
}
```

Utskrift på fil

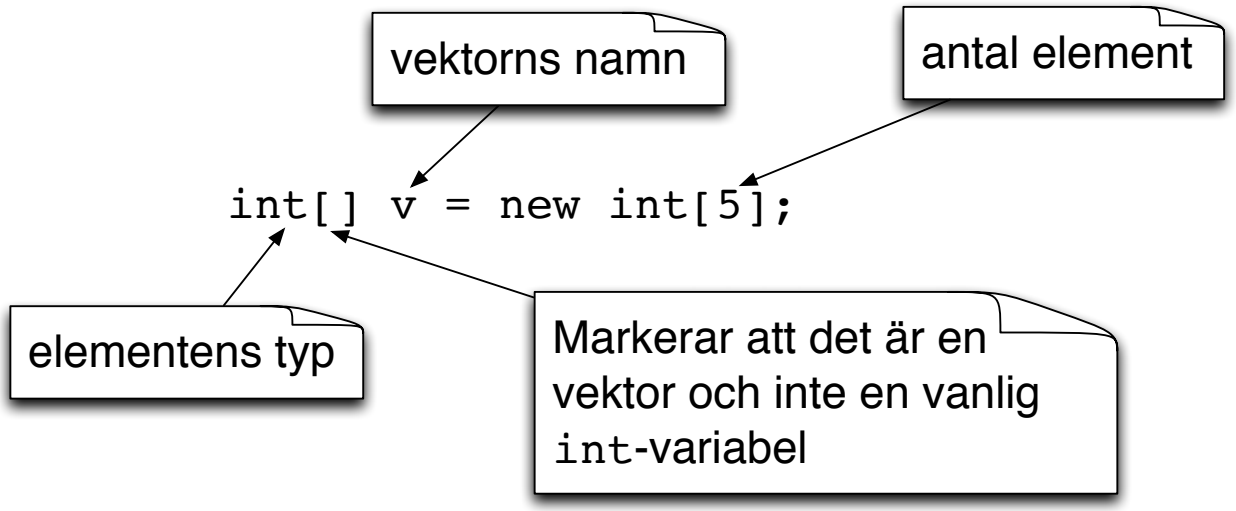
Exempel

```
public class PrintSquareRoots {
    public static void main(String[] args) {
        PrintWriter out = null;
        try {
            out = new PrintWriter(new File("utdata.txt"));
        } catch (FileNotFoundException e) {
            System.out.println("Filen kunde inte öppnas");
            System.exit(1);
        }
        for (int i = 1; i <= 100; i++) {
            out.println(Math.sqrt(i));
        }
        out.close();
    }
}
```

Vektorer

Repetition av begrepp

- Deklarera och skapa vektor:



- Använda vektor:

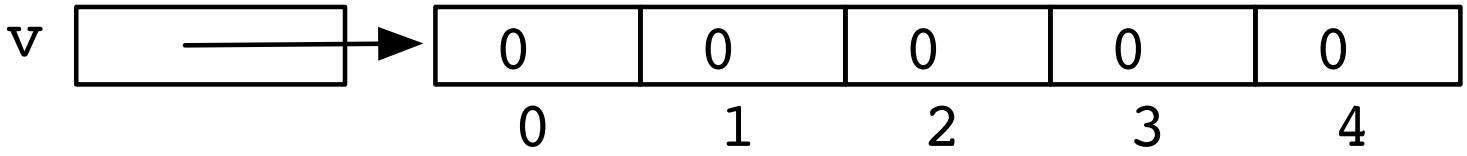
```
for (int i = 0; i < v.length; i++) {
   v[i] = scan.nextInt();
}
```

index

antal element

Vektorer är objekt

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

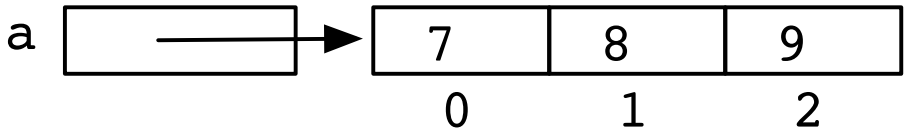


Vektorer är objekt

Referenstilldelning

Vad händer i detta exempel? Hur många vektorer skapas? Vad refererar `b` till? Komplettera bilden.

```
int[] a = new int[3];
a[0] = 7;
a[1] = 8;
a[2] = 9;
int[] b = a;
```

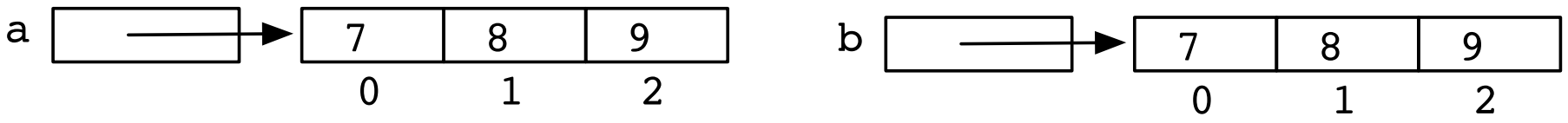


Kopiera vektorer

Exempel

Kopiera a, dvs. skapa en *ny* vektor med samma innehåll som a.

```
int[] a = new int[3];
a[0] = 7;
a[1] = 8;
a[2] = 9;
int[] b = new int[a.length];
for (int i = 0; i < a.length; i++) {
    b[i] = a[i];
}
```



Genvägar

Överkurs, men praktiskt

- Skapa en vektor när man vet innehållet:

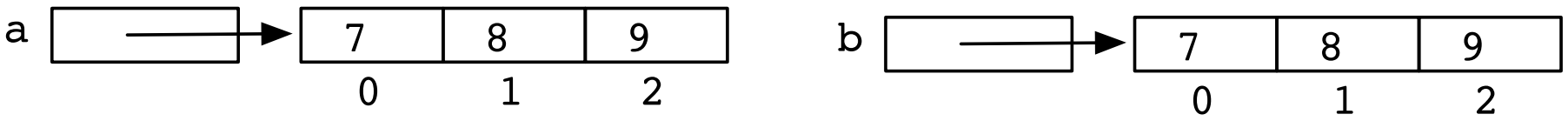
```
int[] a = {7, 8, 9};
```

- Kopiera en vektor till en ny vektor med längden a.length:

```
int[] b = Arrays.copyOf(a, a.length);
```

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

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



Lagra objekt i vektorer

Exempel tärningar

- Exempel: program som skapar 5 tärningar.

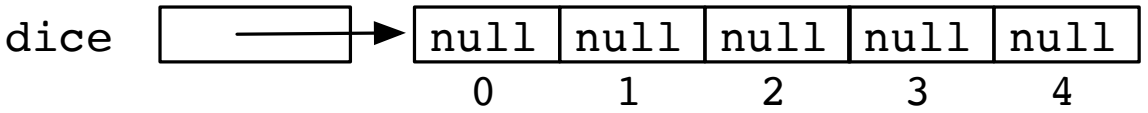
```
public class FiveDice {
    public static void main(String[] args) {
        Die[] dice = new Die[5];

        /** Skapa tärningarna */
        for (int i = 0; i < dice.length; i++) {
            dice[i] = new Die();
        }

        /** Kasta tärningarna */
        for (int i = 0; i < dice.length; i++) {
            dice[i].roll();
            System.out.println("Tärning pos " + i +
                    " visar " + dice[i].getDots());
        }
    }
}
```

Lagra objekt i vektorer

Deklarera och skapa vektorn

- Vektorn `dice` med plats för fem `Die`-objekt:

```
Die[] dice = new Die[5];
```

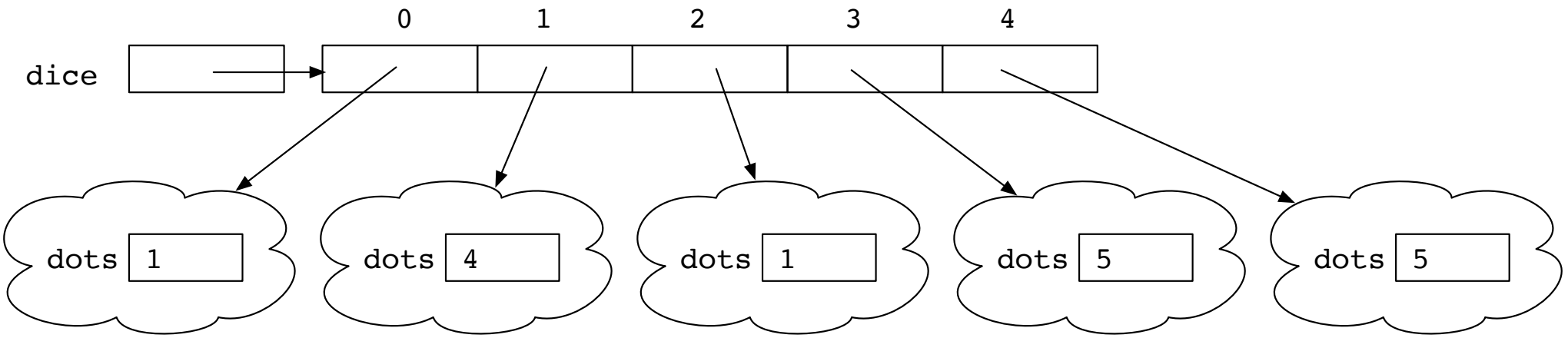


- Varje vektorelement är en referensvariabel av typen `Die`.
- Observera att vi inte har skapat några tärningsobjektobjekt ännu, bara en vektor med plats för fem tärningar.

Lagra objekt i vektorer

Skapa objekten

```
for (int i = 0; i < dice.length; i++) {
    dice[i] = new Die();
}
```



Klassen Triangle

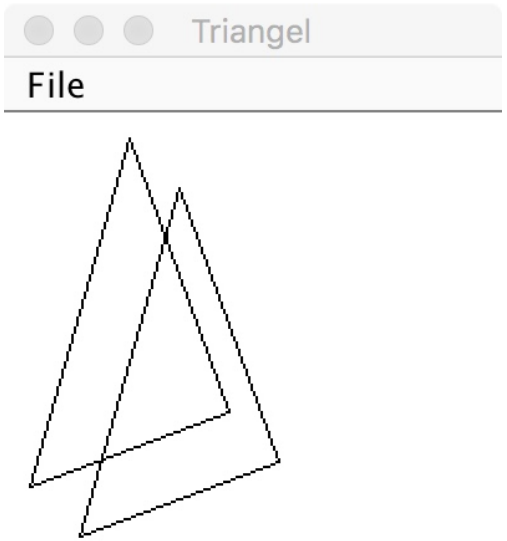
Klassen `Triangle` har följande specifikation:

```
/** Skapar en triangel med hörnpunkterna x1 y1, x2 y2 och
    x3 y3. */
Triangle(int x1, int y1, int x2, int y2, int x3, int y3);

/** Flyttar triangeln dx i x-led och dy i y-led. */
 void move(int dx, int dy);

/** Ritar triangeln i fönstret w. */
public void draw(SimpleWindow w);
```

Implementera klassen.

Program som använder klassen Triangle

```
public class TriangleExample {
    public static void main(String[] args) {
        Triangle t = new Triangle(10, 150, 50, 10, 90, 120);
        SimpleWindow w = new SimpleWindow(200, 200, "Triangel");
        t.draw(w);
        t.move(20, 20);
        t.draw(w);
    }
}
```



Lagra objekt i vektorer

Exempel triangel

- Hur ska vi hålla reda på hörnen?
  - 6 st attribut med typen `int`: `x1`, `y1`, `x2`, `y2`, `x3`, `y3`?
  - 3 st attribut med typen `Point`: `p1`, `p2`, `p3`?
  - en vektor med tre punkter?

- Vi väljer den tredje varianten. Det finns en färdig klass `Point` i paketet `java.awt`. Vi kommer att använda dessa metoder:

```
double getX();
double getY();
void translate(int dx, int dy)  // flyttar punkten
                                // dx längs x-axeln och
                                // dy längs y-axeln
```

Klassen Triangle

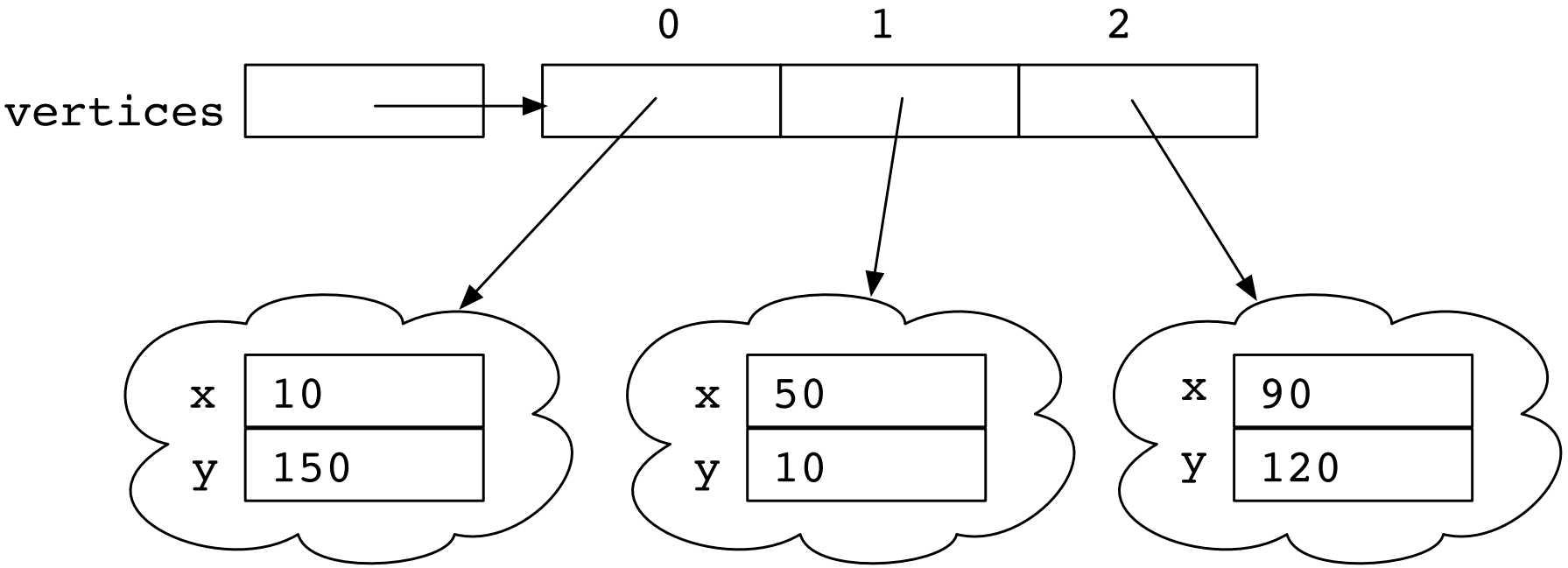
attribut, konstruktor och metoden move

```
public class Triangle {
    private Point[] vertices;

    /** Skapar en triangel med hörnpunkterna x1 y1, x2 y2 och x3 y3. */
    public Triangle(int x1, int y1, int x2, int y2, int x3, int y3) {
        vertices = new Point[3];
        vertices[0] = new Point(x1, y1);
        vertices[1] = new Point(x2, y2);
        vertices[2] = new Point(x3, y3);
    }

    ...
```

- Observera att både vektorn och punkterna skapas i konstruktorn i det här exemplet.

Lagra objekt i vektorer

Attributet vertices



Klassen Triangle

Metoderna draw och move

```
/** Flyttar triangeln dx i x-led och dy i y-led. */
public void move(int dx, int dy) {
    for (int i = 0; i < vertices.length; i++) {
        vertices[i].translate(dx, dy);
    }
}

/** Ritar triangeln i fönstret w. */
public void draw(SimpleWindow w) {
    // Flytta först till sista punkten så att vi därefter
    // kan loopa igenom punkt 0-2 i ordning
    w.moveTo((int) Math.round(vertices[2].getX()),
             (int) Math.round(vertices[2].getY()));
    for (int i = 0; i < vertices.length; i++) {
        w.lineTo((int) Math.round(vertices[i].getX()),
                 (int) Math.round(vertices[i].getY()));
    }
}
```

Diskutera

- Vi vill lägga till en metod i klassen `Triangle` som undersöker om en punkt med koordinaterna x, y ingår bland triangelns hörnpunkter.
- Börja skissa på metoden:
  - Hitta på ett lämpligt namn
  - Vilka parametrar ska metoden ha?
  - Returtyp?
  - Börja skissa på algoritm och/eller programkod

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.
- *Algoritm*:

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

Linjärsökning

Exempel 1

- Lägg till en metod som undersöker om det finns någon hörnpunkt med koordinaterna x, y.

```
public boolean hasVertex(int x, int y) {
    for (int i = 0; i < vertices.length; i++) {
        if (vertices[i].getX() == x && vertices[i].getY() == y) {
            return true;
        }
    }
    return false;
}
```

Håll koll på typerna

- Kontrollera att typerna stämmer vid tilldelningar och jämförelser för att undvika fel.

	namn	typ
`index`	`i`	`int`
`vektorn med punkter`	`vertices`	`Point[]`
`en punkt`	`vertices[i]`	`Point`
`punktens x-koordinat`	`vertices[i].getX()`	`double`
`punktens y-koordinat`	`vertices[i].getY()`	`double`

Se upp med `return`

```
public boolean hasVertex(int x, int y) {
    for (int i = 0; i < vertices.length; i++) {
        if (vertices[i].getX() == x && vertices[i].getY() == y) {
            return true;
        } else {
            return false;
        }
    }
}
```

- I den felaktiga koden ovan undersöker man bara `vertices[0]` och returnerar sedan direkt `true` eller `false`.
- Man ska fortsätta leta tills man hittat en punkt med matchande x- och y-koordinat eller tills alla punkter är genomsökta.

return och break

- Med `return` avbryter man exekveringen av en metod.
- Med `break` avbryter man en loop.

```
while (true) {
    ...
    if (...) {
        break;  // hoppar ut ut loopen
    }
}
```

  - Ska användas med försiktighet så att koden inte blir för svårläst.

Linjärsökning

Exempel 2

- Lägg till en metod som undersöker om punkten p matchar någon av triangelns hörnpunkter.

```
public boolean hasVertex(Point p) {
    for (int i = 0; i < vertices.length; i++) {
        if (vertices[i].getX() == p.getX() &&
                vertices[i].getY() == p.getY()) {
            return true;
        }
    }
    return false;
}
```

- Alternativ lösning: I klassen `Point` finns det en metod `equals` som jämför två punkters innehåll:

```
... if (vertices[i].equals(p)) { ...
```

Linjärsökning

Med `while`-sats utan avbrott mitt i

- *Algoritm*:

```
i = "platsen för det första elementet"
while ("fler element kvar att söka igenom" &&
    "elementet på plats i inte är det vi söker")
    i = platsen för nästa element
```

- *Exempel*:

```
public boolean hasVertex(int x, int y) {
    int i = 0;
    while (i < vertices.length && (vertices[i].getX() != x ||
                                   vertices[i].getY() != y)) {
        i++;
    }
    return i < vertices.length;
}
```

Linjärsökning

Med `while`-sats – undvik fel

- Ordningen mellan delvillkoren är viktig.
  - I ett logiskt uttryck med flera delvillkor exevkeras inte mer än vad som behövs för att beräkna det logiska uttryckets värdet.
  - Om `i < vertices.length` är `false` undersöks inte det andra deluttrycket. Man undviker därför att använda ett för stort index i `vertices`.

- Efter while-satsen vill man veta om det man sökte efter fanns eller ej.
  - Jämför `i` med antal element för att avgöra detta:

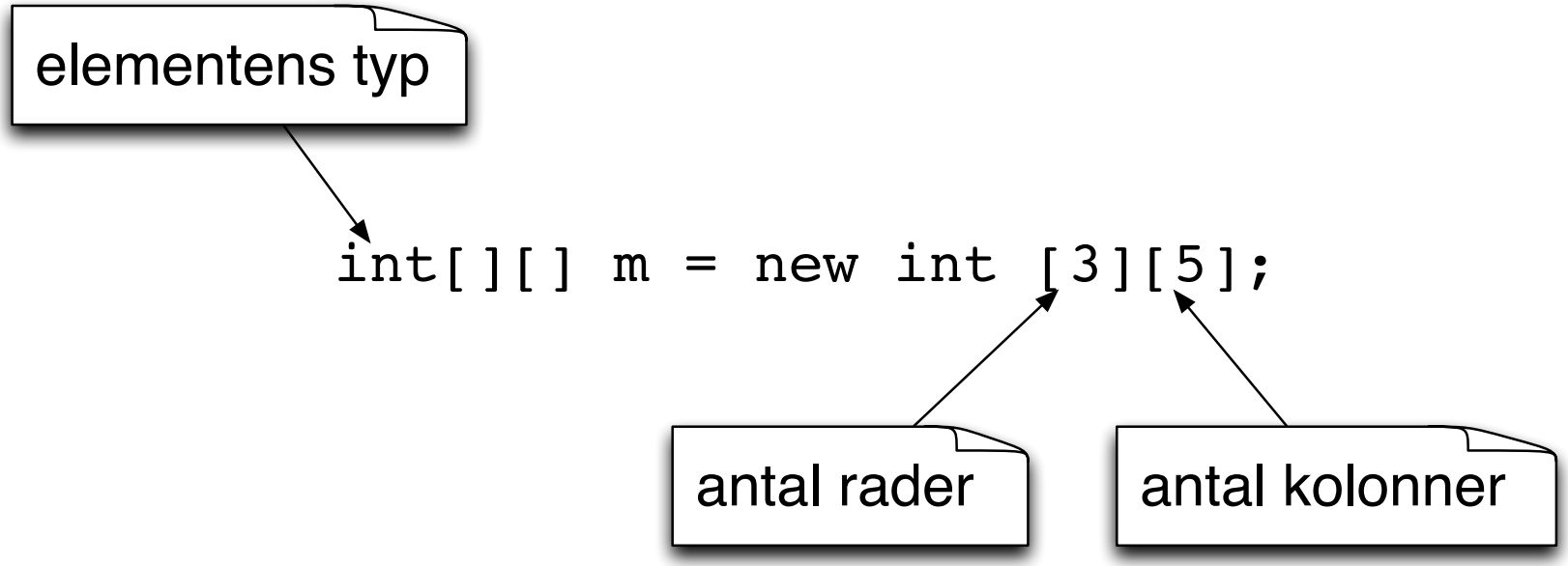
```
if (i < vertices.length) {
     return true
 } else {
    return false;
}
```

  - Om det sökta inte finns har `i` fått värdet `vertices.length`. Därför kan man inte använda `vertices[i]` i villkoret i if-satsen.

Matriser

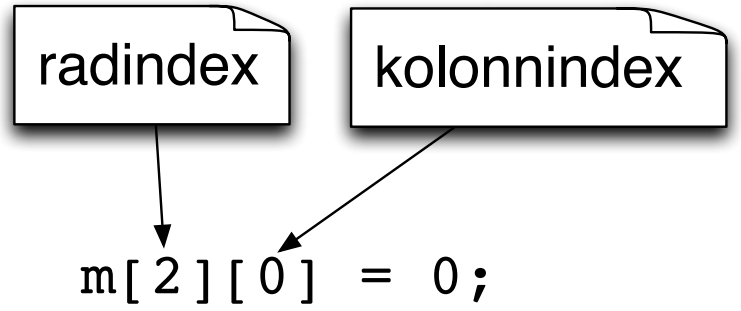
	kolonn 0	kolonn 1	kolonn 2	kolonn 3	kolonn 4
rad 0	7	9	123	41	1
rad 1	22	-18	12	3	-2
rad 2	11	16	-4	0	1

Deklarera och skapa matriser



Använda matriser

- Att använda ett matriselement:



- Man kan ta reda på antal rader:

```
m.length
```

och antal kolumner på raden i:

```
m[i].length
```

Övning

Matris

Deklarera och skapa en matris som representerar ett schackbräde med 8 x 8 rutor. På rutorna ska man kunna placera schackpjäser som beskrivs av en klass `ChessPiece` (som vi antar finns).

Exempel

Matris

Beräkna och skriv ut radsummorna i en 3 x 5-matris.

```
för varje rad
    för varje element på raden
        behandla elementet
```

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

Övning

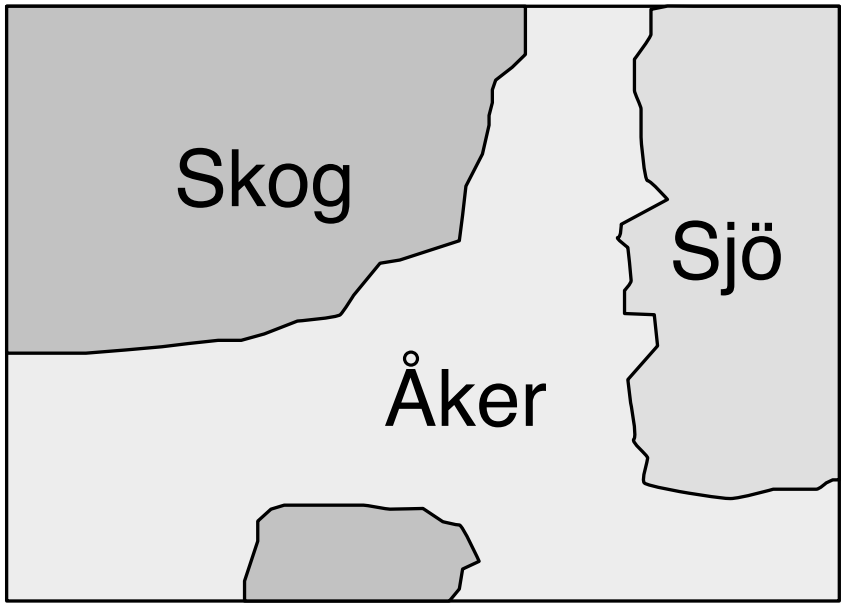
Matris

Antag att vi har en matris `booked` av typen `boolean[][]` som ska användas för att hålla reda på bokade platser i en biosalong. Varje matriselement motsvarar en plats i salongen och har värdet `true` om platsen är bokad, i annat fall värdet `false`.

Skriv satser som räknar antal bokade platser.

Programexempel: rasterdata

- En kartas geometriska innehåll kan lagras i form av rasterdata. Det geometriska innehållet lagras som en matris där varje element motsvarar en ruta (t.ex. 50 x 50 kvadratmeter).
- I vårt exempel innehåller varje matriselement ett heltal som representerar markanvändning för den ruta som elementet motsvarar i verkligheten. Talet 1 betyder skog, 2 betyder åker och 3 betyder sjö.



1	1	1	1	2	3	3
1	1	1	1	2	3	3
1	1	1	2	2	3	3
2	2	2	2	2	3	3
2	2	1	1	2	2	2

Programexempel: rasterdata

```
public class RasterData {
    public static void main(String[] args) {
        // 1 = skog, 2 = åker, 3 = sjö
        int[][] rasterdata = {{1, 1, 1, 1, 2, 3, 3},
                              {1, 1, 1, 1, 2, 3, 3},
                              {1, 1, 1, 2, 2, 3, 3},
                              {2, 2, 2, 2, 2, 3, 3},
                              {2, 2, 1, 1, 2, 2, 2}};

        // Lägg till satser som beräknar och
        // skriver ut andel åker

    }
}
```

Programexempel: rasterdata

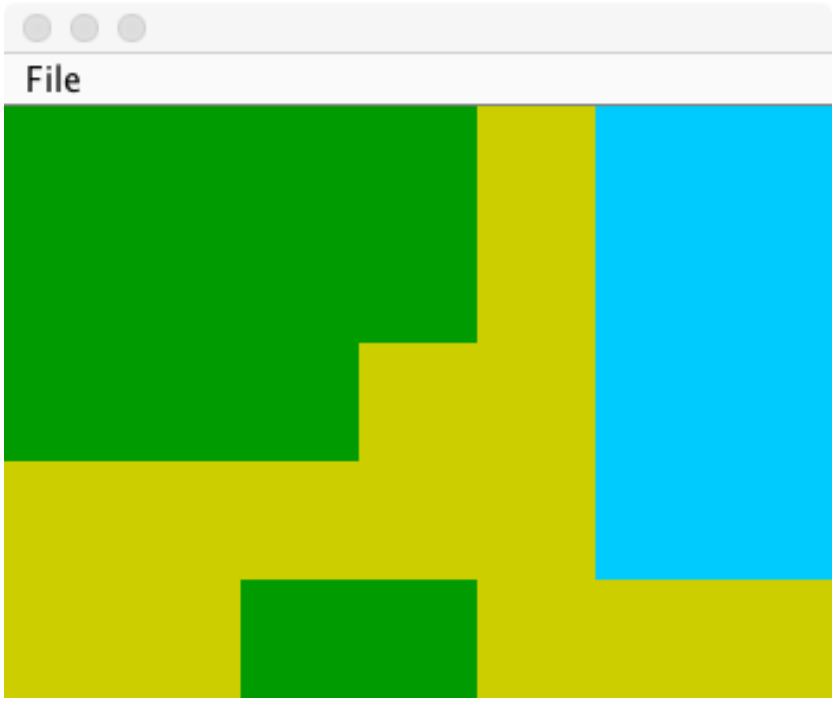
Andel åker

```
int nbrFields = 0;
for (int row = 0; row < rasterdata.length; row++) {
    for (int col = 0; col < rasterdata[row].length; col++) {
        if (rasterdata[row][col] == 2) {
            nbrFields++;
        }
    }
}
double share = (double) nbrFields /
                    (rasterdata.length * rasterdata[0].length);
System.out.println("Andelen åker är " + share);
```

Programidé: rasterdata

karta

- Programmet från föregående bilder kan modifieras så att man får en bild av markanvändningen.
- Använd t.ex. klassen `Graphics` från lab 5 och rita block i olika färg för skog, åker och sjö.



Checklista

Exempel på vad du ska kunna

- Förklara begreppen datastruktur, vektor, matris.
- Deklarera, skapa och använda vektorer och matriser.
- Formulera algoritmer och programkod för att söka i en vektor (linjärsökning).
- Läsa data från en textfil
- Skriva data på en textfil