EDAA20
Programmering och databaser

Föreläsning 10 – Linjärsökning och matriser

2023-09-26, Niklas Fors

Från **Kursplanen** (se Moodle-sida)

Programmering

	Läsvecka 1	Läsvecka 2	Läsvecka 3	Läsvecka 4	Läsvecka 5	Läsvecka 6	Läsvecka 7	
Mån	F1	F3	F5	F7	F9	F11	F13	
Tis	F2	F4	F6	F8	F10	F12	F14	
Ons	L1	L3	L4	L6	L8	L	L11	
Fre	L2	L	L5	L7	L9	L10	L	Instuderings-vecka
Ons/Tors	R	R	R	R	R	R	R	

F föreläsningar, L datorlaborationer (obligatoriska), R resurstid

Laboration 8	spara data på textfil, polygoner
Laboration 9	matriser, Memory-spel . . . .

Tenta, måndag 23 oktober

Klassen `Triangle`

Klassen `Triangle` har följande specifikation:

```
/** Skapar 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 relativt med avstånden dx, dy */
void move(int dx, int dy);

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

Reptition: konstruktorn i klassen `Triangle`

```
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);
  }

  ...
}
```

Konstruktorn skapar både vektorn och `Point`-objekten

Sökning

Vi ska lägga till en metod för att kontrollera om koordinaterna *x,y* är någon av triangelns hörnpunkter.

Frågor om metoden:

- Namn?
- Returtyp?
- Parametrar?

Exempel på användning:

```
Triangle t = new Triangle(10, 150, 50, 10, 90, 120);
t.move(5, 5);

if (t.hasVertex(55, 15)) {
  System.out.println("(55, 15) är en hörnpunkt");       // Skrivs ut
} else {
  System.out.println("(55, 15) är INTE en hörnpunkt");
}
```

Linjärsökning – algoritm

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

Indata: sekvens av element, eftersökt element
Utdata: sant om elementet finns, annars falskt
Pseudokod:

```
för varje element x i sekvensen
    om x är det sökta elementet
        avbryt sökningen och returnera sant
returnera falskt
```

Linjärsökning – implementation

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

  /** Returnerar true om koordinaten x,y är någon av hörnpunkterna,
    * annars false. */
  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;
  }
}
```

Vanligt fel: för tidig `return`!

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

  /** Returnerar true om koordinaten x,y är någon av hörnpunkterna,
    * annars false. */
  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;
      }
    }
  }
}
```

Koden undersöker *enbart* första punkten,
och avbryter sedan exekveringen av metoden.

Detta är ett vanligt fel som man ska undvika.

Övning: vilka typer har följande uttryck?

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

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

Uttryck	Typ?
i	
x	
y	
vertices	
vertices[i]	
vertices[i].getX()	
vertices[i].getY()	

Övning: vilka typer har följande uttryck?

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

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

Uttryck	Typ
i	int
x	int
y	int
vertices	Point[]
vertices[i]	Point
vertices[i].getX()	int
vertices[i].getY()	int

Håll koll på typerna!

Övning: vilka typer har följande uttryck?

Beskrivning	Uttryck	Typ
Index	`i`	`int`
X-koordinat som parameter	`x`	`int`
Y-koordinat som parameter	`y`	`int`
Vektorn med punkter	`vertices`	`Point[]`
En punkt i vektorn	`vertices[i]`	`Point`
X-koordinat för en punkt	`vertices[i].getX()`	`int`
Y-koordinat för en punkt	`vertices[i].getY()`	`int`

Alternativ: sökning med **break** istället för **return**

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

  /** Returnerar true om koordinaten x,y är någon av hörnpunkterna,
    * annars false. */
  public boolean hasVertex(int x, int y) {
    boolean found = false;
    for (int i = 0; i < vertices.length; i++) {
      if (vertices[i].getX() == x && vertices[i].getY() == y) {
        found = true;
        break;
      }
    }
    return found;
  }
}
```

Satsen `break`

En `return`-sats avbryter exekveringen av en metod

- *(Returnerar möjligtvis ett värde)*

En `break`-sats avbryter exekveringen av *närmaste* `for`/`while`-loop

```
while (...) {
  ...
  if (...) {
    break;   // avbryter exekveringen av while-loopen
  }
}
```

(har man två nästlade loopar avbryter `break` den innersta)

Matris – vektor av vektorer

Skapa en 3x4-matris:

```
int[][] m = new int[3][4];
m[0][0] = 4;
m[0][1] = 6;
...
```

Rad Kolumn

	kolumn 0	kolumn 1	kolumn 2	kolumn 3
rad 0	4	6	5	6
rad 1	2	7	4	4
rad 2	2	1	8	5

Matriser

Med givna startvärden:

```
int[][] m = {
  {4, 6, 5, 6},
  {2, 7, 4, 4},
  {2, 1, 8, 5}
};
```

	kolumn 0	kolumn 1	kolumn 2	kolumn 3
rad 0	4	6	5	6
rad 1	2	7	4	4
rad 2	2	1	8	5

Matriser

Skapa matris med 3 rader och 5 kolumner:

```
int[][] m = new int[3][5];
```

Tilldela elementet på rad 2 och kolumn 3 ett nytt värde:

```
m[2][3] = 42;
```

Antalet rader:

```
m.length
```

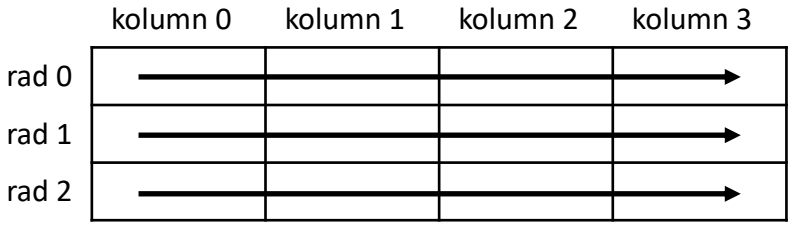
Antalet kolumner på rad `i`:

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

Behandla alla element

Pseudokod:

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



Först behandlas alla element på första raden, sedan alla element på andra raden, osv

Matrisexempel 1

Beräkna summan av elementen och skriv ut:

```
int sum = 0;
for (int i = 0; i < m.length; i++) {
  for (int j = 0; j < m[i].length; j++) {
    sum += m[i][j];
  }
}
System.out.println(sum); // 54
```

	kolumn 0	kolumn 1	kolumn 2	kolumn 3
rad 0	4	6	5	6
rad 1	2	7	4	4
rad 2	2	1	8	5

Matrisexempel 2

Beräkna summan för ***varje rad*** och skriv ut den

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

	kolumn 0	kolumn 1	kolumn 2	kolumn 3	
rad 0	4	6	5	6	21
rad 1	2	7	4	4	17
rad 2	2	1	8	5	16

Övning

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.

Tips: ta inspiration från tidigare exempel

```
int sum = 0;
for (int i = 0; i < m.length; i++) {        // För varje rad
  for (int j = 0; j < m[i].length; j++) {   // För varje element på raden
    sum += m[i][j];
  }
}
```

Lösning

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.

```
int numBooked = 0;
for (int i = 0; i < booked.length; i++) {
  for (int j = 0; j < booked[i].length; j++) {
    if (booked[i][j]) {
      numBooked++;
    }
  }
}
```

Programexempel: 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

- 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.

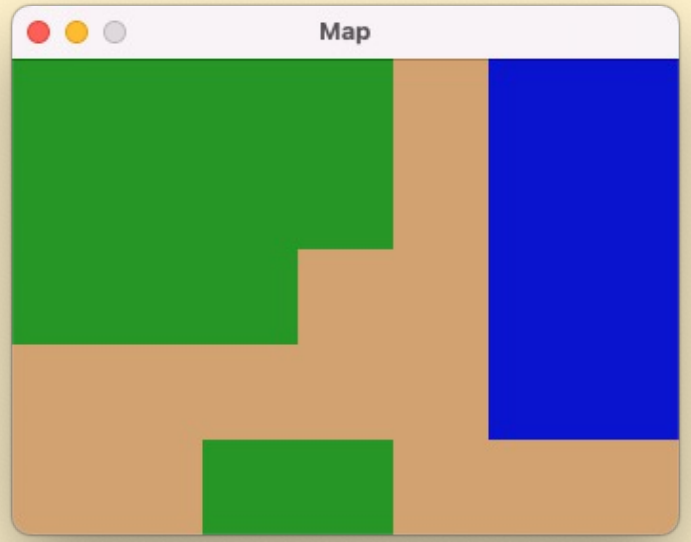
```java
public class RasterData {
  public static void main(String[] args) {
    // 1 = skog, 2 = åker, 3 = sjö
    int[][] raster = {{1, 1, 1, 1, 2, 3, 3},
                      {1, 1, 1, 1, 2, 3, 3},
                      {1, 1, 1, 2, 2, 3, 3},
                      {2, 2, 2, 2, 2, 3, 3},
        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





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




  }
}
```

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

    int nbrFields = 0;
    for (int i = 0; i < raster.length; i++) {
      for (int j = 0; j < raster[i].length; j++) {
        if (raster[i][j] == 2) {
          nbrFields++;
        }
      }
    }
```

Tvinga flyttalsdivision

```java
    double share = (double)nbrFields / (raster.length * raster[0].length);
    System.out.println("Andel åker: " + share);
  }
}
```

```java
public class RasterDataVisualization {
  public static final Color FIELD = new Color(204, 153, 102);
  public static final Color WATER = new Color(12, 20, 200);
  public static final Color FOREST = new Color(34, 139, 34);

  public static void main(String[] args) {
    int[][] raster = ... som föregående bild ...
    Graphics g = new Graphics(raster[0].length, raster.length, 50);
    for (int i = 0; i < raster.length; i++) {
      for (int j = 0; j < raster[i].length; j++) {
        if (raster[i][j] == 1) {
          g.block(j, i, FOREST);
        } else if (raster[i][j] == 2) {
          g.block(j, i, FIELD);
        } else if (raster[i][j] == 3) {
          g.block(j, i, WATER);
        }
      }
    }
  }
}
```

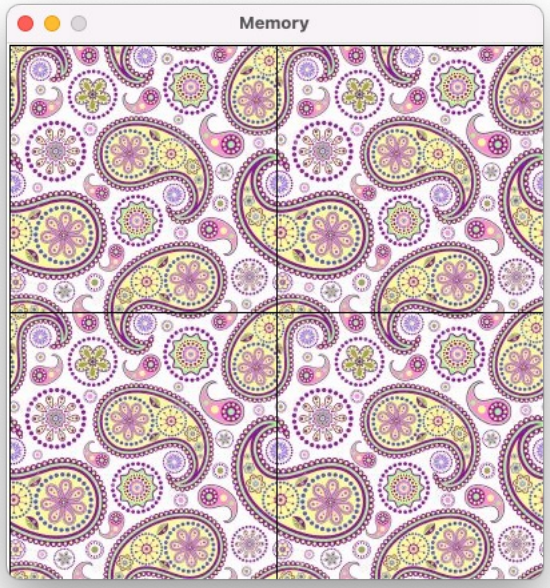


Visualisering med hjälp av klassen `Graphics` från labb 5.
Denna fil kan placeras i labb 5-projektet.

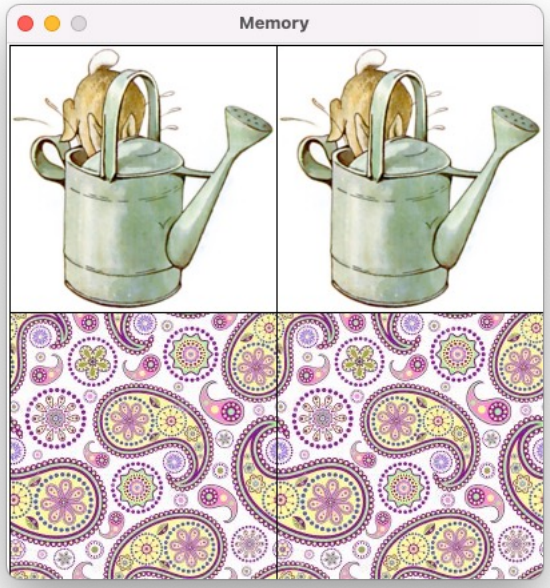
Memory-spel med 2x2-kort visas i föreläsningsbilderna
I labben har man 4x4-kort (samma princip gäller)

Laboration 9

Laboration 9 handlar om att implementera spelet Memory:

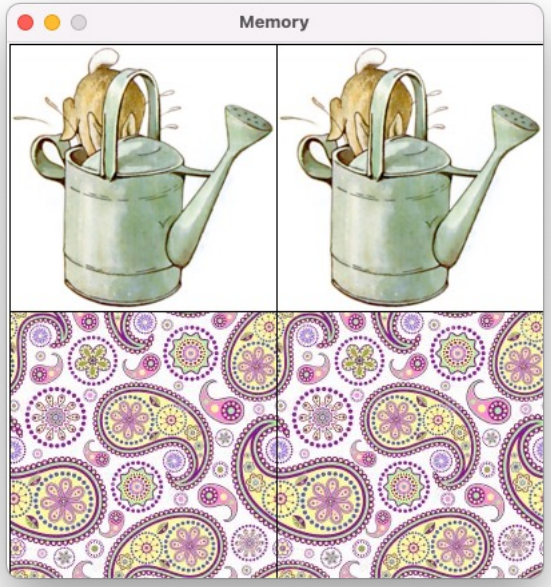


Från början visas baksidorna



Användaren vänder upp två kort i taget.
Korten får ligga kvar om de har samma framsida,
annars vänds de tillbaka.

Brädets representation

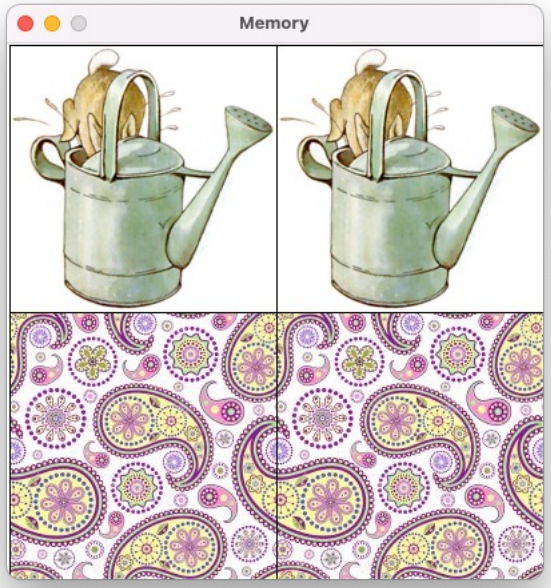


Brädet kommer ihåg:

- Vilka kort som har framsidan uppåt: `boolean[][]`
- Vilka kort som ligger var: `MemoryCardImage[][]`

`MemoryCardImage` har en fram- och baksida

Brädets representation



Matris över vilka kort som har framsidan uppåt

`true`	`true`
`false`	`false`

Matris över vilka kort som ligger var

`MemoryCardImage`

`front`	`"can.jpg"`
`back`	`"back.jpg"`

`MemoryCardImage`

`front`	`"friends.jpg"`
`back`	`"back.jpg"`

Två kort med samma framsida representeras med ***två referenser*** i matrisen till ***ett*** **`MemoryCardImage`-objekt**

Utplacering av kort

för varje framsida
    skapa ett `MemoryCardImage`-objekt för framsidan
    placera ut objektet på 2 slumpmässiga men lediga platser i matrisen

Utplacering av kort (1/6)

Matris över vilka kort som ligger var

`null`	`null`
`null`	`null`

Skapa en matris med tomma platser (`null`) för kort

Utplacering av kort (2/6)

Matris över vilka kort som ligger var

null	null
null	null

MemoryCardImage

front	"can.jpg"
back	"back.jpg"

Skapa MemoryCardImage-objekt för framsidan "can.jpg"

Utplacering av kort (3/6)

Matris över vilka kort som ligger var

null	
null	null

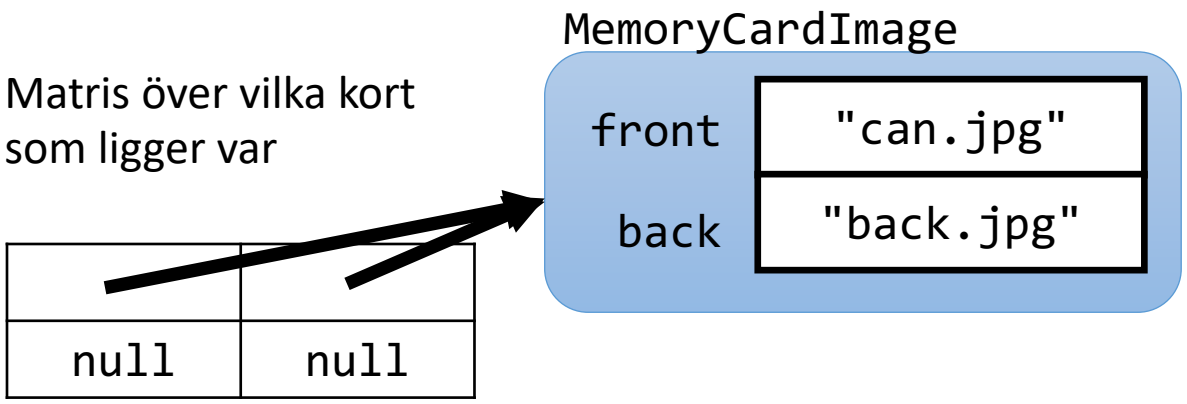
MemoryCardImage

front	"can.jpg"
back	"back.jpg"

Slumpa fram första platsen:
rad 0, kolumn 1

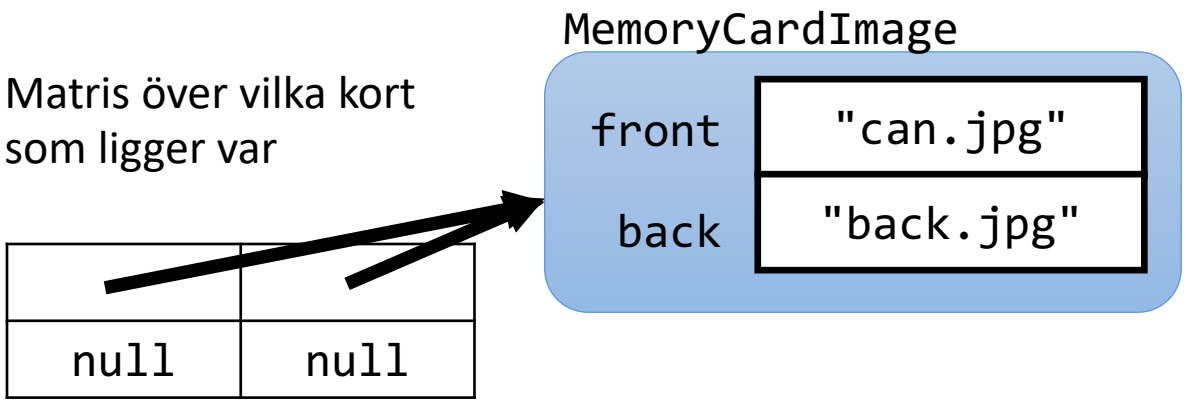
Platsen är ledig $\Longrightarrow$ placera kortet

Utplacering av kort (4/6)



Slumpa fram andra platsen:
rad 0, kolumn 0

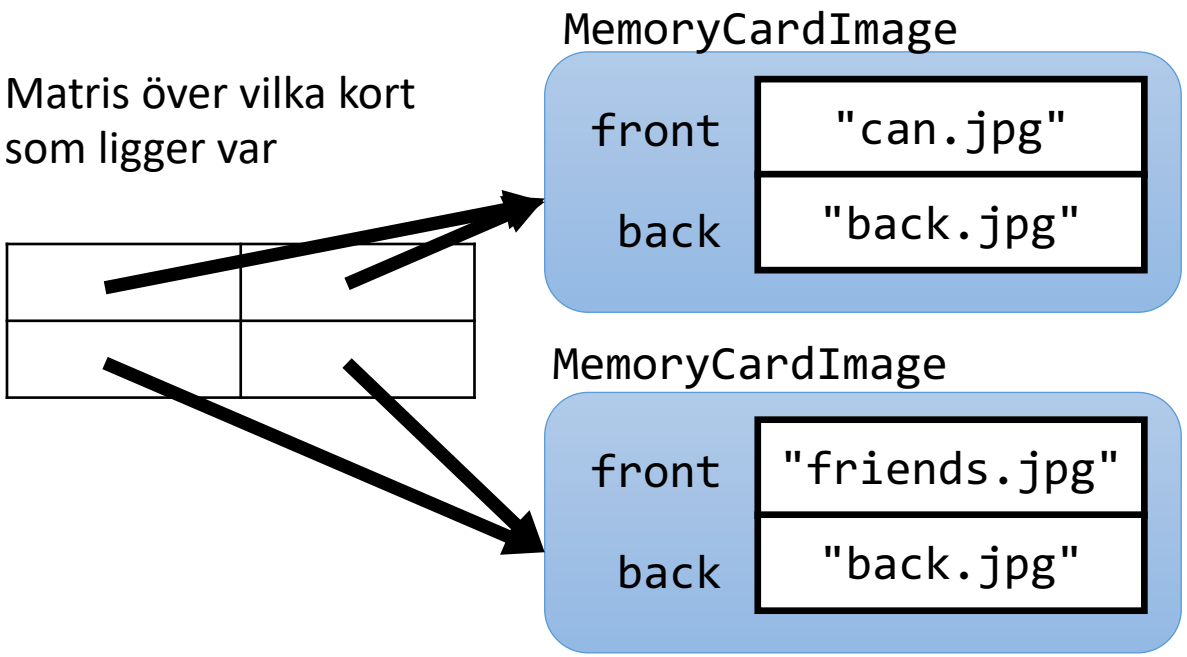
Utplacering av kort (5/6)



Om plats är upptagen (ej `null`)
⟹ slumpa ny plats

Utplacering av kort (6/6)

Samma sak görs för framsidan "friends.jpg".



Ledtrådar

Film om några ledtrådar till Memory-labben:
https://lu.instructuremedia.com/embed/27a87a5f-0ce3-4522-981a-256c12cc9377

Checklista

- 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