F7 - Arrayer

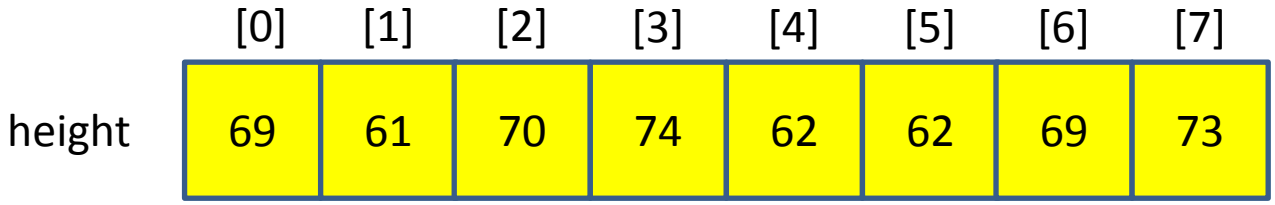
ID1004 Objektorienterad programmering

Fredrik Kilander – fki@kth.se

Array, arrayer, arrayen

- En array är en lista av variabler av samma typ
- Arrayen har en fast längd (antal element)
- Enskilda element nås med numeriskt index

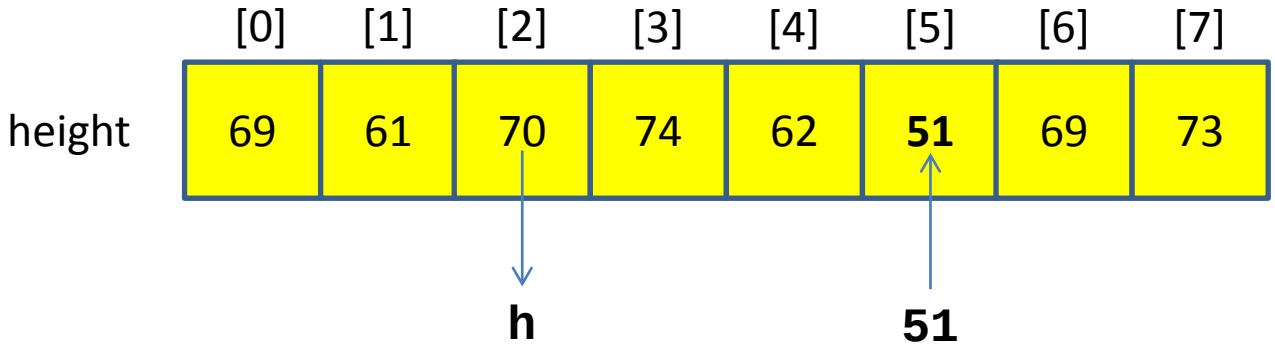
```
int [] height = new int[8];
```



Array, arrayer, arrayen

- Tilldelning: **`height[5] = 51;`**
- Läsning: **`int h = height[2];`**
- Variabeln h får värdet 70

```
int [] height = new int[8];
```



Arrayer är objekt

- En array-variabel är en referensvariabel
- Datatypen array of int: **`int []`**
- Variabelns namn: **`height`**
- Instansen skapas med new: **`new int[8];`**
- Därefter kan antalet element inte ändras

```
int [] height = new int[8];

float [] temps; // Ingen initiering, null
```

Arraytyp och elementtyp

- Array-variabeln är av *arraytyp*
- Alla element i en array har samma datatyp
- Arrayen har endast en *elementtyp*

```
String [] messages;
```

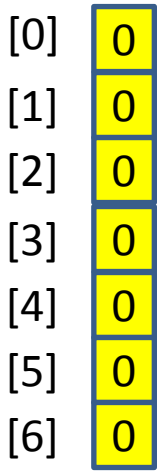
Variabeln messages är av arraytyp

Elementtypen är String

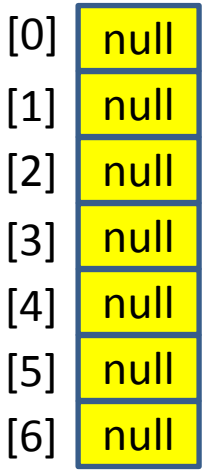
Arrayer

- En array skapas med alla element
- En primitiv elementtyp initieras till noll
- En elementtyp som refererar initeras till null

```
long [] times = new long[7];
```




```
String[] message = new String[7];
```



Arrayer - index

- Indexering utanför arrayen bestraffas med
  - **`ArrayIndexOutOfBoundsException`**
- Lägsta index: 0
- Högsta index: Array.length - 1

```
long [] times = new long[7];
```

[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
[5]	0
[6]	0

```
String[] message = new String[7];
```

[0]	null
[1]	null
[2]	null
[3]	null
[4]	null
[5]	null
[6]	null

Arrayer - index

- En array är ett objekt
- Arrayens längd finns i Array.length, t ex:
  - **`height.length`**
  - **`messages.length`**
- Om arrayvariabeln inte refererar till ett array-objekt (är null) så blir det
  - **`NullPointerException`**

Gå igenom en array - läsning

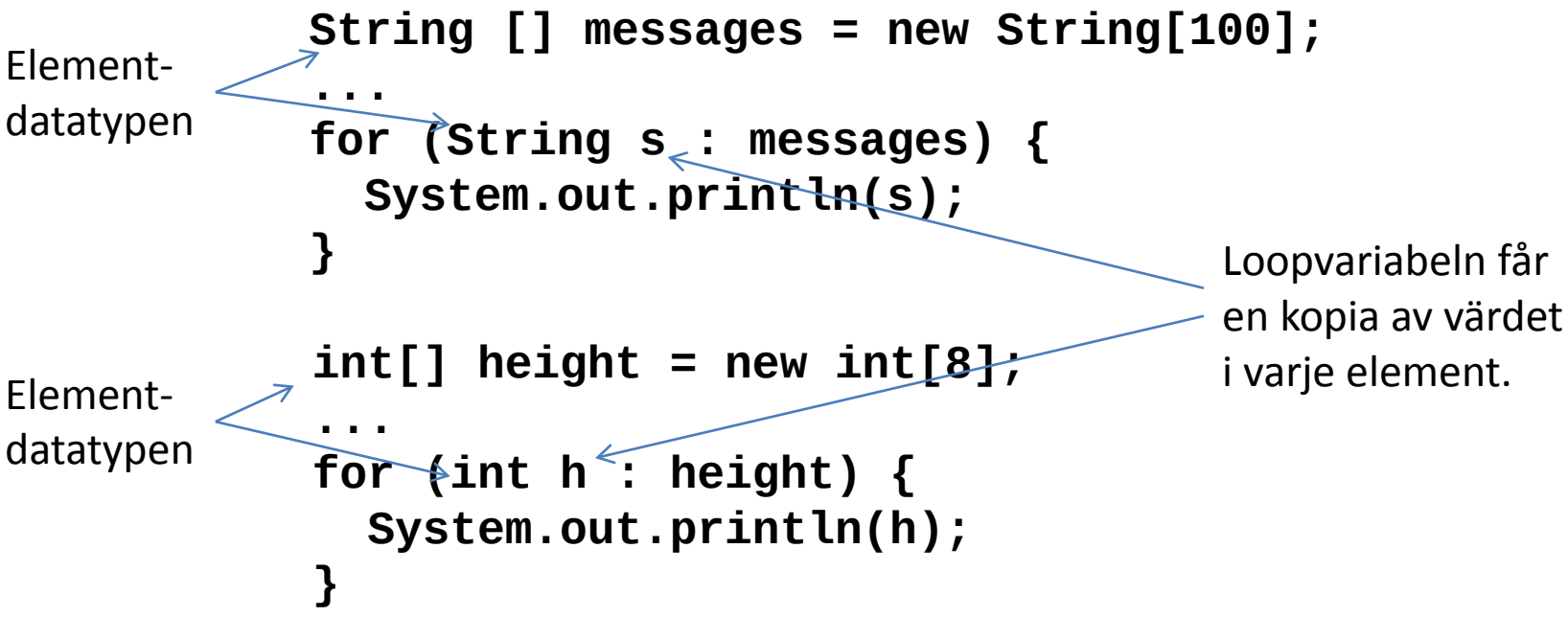
- Alla element i en array kan läsas från första till sista med for-each-loopen:

```
String [] messages = new String[100];
...
for (String s : messages) {
  System.out.println(s);
}

int[] height = new int[8];
...
for (int h : height) {
  System.out.println(h);
}
```

Gå igenom en array - läsning

- Alla element i en array kan läsas från första till sista med for-loopen:



Gå igenom en array – läsning/skrivning

- För skrivning måste arrayen indexeras:

```
int[] height = new int[8];
...
int offset = -3;
...
for (int i = 0; i < height.length; i++) {
  height[i] += offset;
}
```

Alternativ deklarationssyntax

- Arraytyper kan deklareras på två sätt
  - `int [] height; // Java-style`
  - `int height [];  // C/C++-style`

- Båda sätten har samma mening

- Java-style förordas för ny Java-kod

Initiering av arrayer

- En array kan initieras vid skapandet
- Alla element måste initieras

```
int [] scores = {87, 98, 69, 87 65, 76};
```


Ingen new behövs

Arrayer som parametrar

- Arrayen är ett objekt
- Referensen till arrayen överförs till parametern
- Metoden kan alltså ändra i arrayen

```
private void swapCards(int [] deck, int i, int j) {
  int temp = deck[i];
  deck[i] = deck[j];
  deck[j] = temp;
}

public void shuffle(int [] deck) {
  Random rnd = new Random();
  for (int n = 0; n < 1000; n++) {
    swapCards(deck, rnd.nextInt(deck.length),
                    rnd.nextInt(deck.length));
  }
}
```

Arrayer av objekt

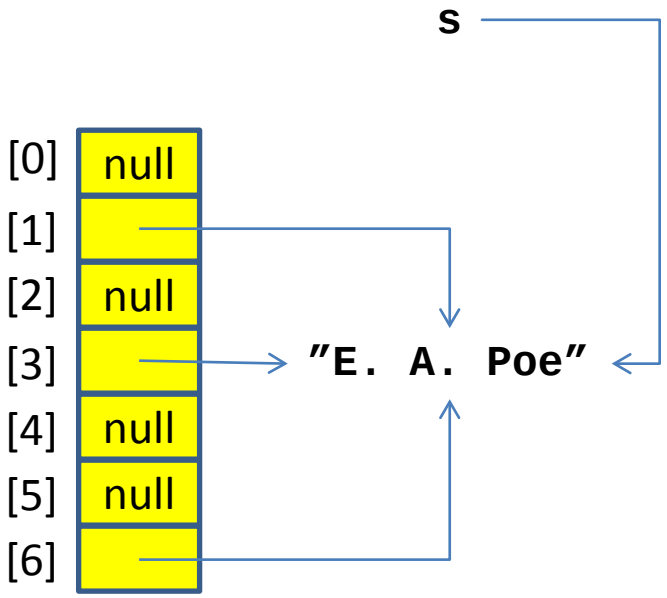
- Varje element är en referens till objekttypen
- Varje element är null från början
- Referensen tilldelas, *inte* objektet

```
String[] message = new String[7];
```

Arrayer av objekt

- Varje element är en referens till objekttypen
- Varje element är null från början
- Referensen tilldelas, *inte* objektet

```
String[] message = new String[7];
String s = ”E. A. Poe”;
message[1] = s;
message[3] = s;
message[6] = s;
```



Arrayer av objekt

- Arrayer av objekt kan också initieras

```
String [] engineMessages =
  {”Fel i motor”, ”Ingen bensin”, ”Lågt batteri”};

Dog [] kennel =
  {new Dog(”Fido”), new Dog(”Karo”), new Dog(”Rex”)};
```

Tvådimensionella arrayer

- Fler dimensioner med fler index
- **`int [][] table = new int[5][10];`**

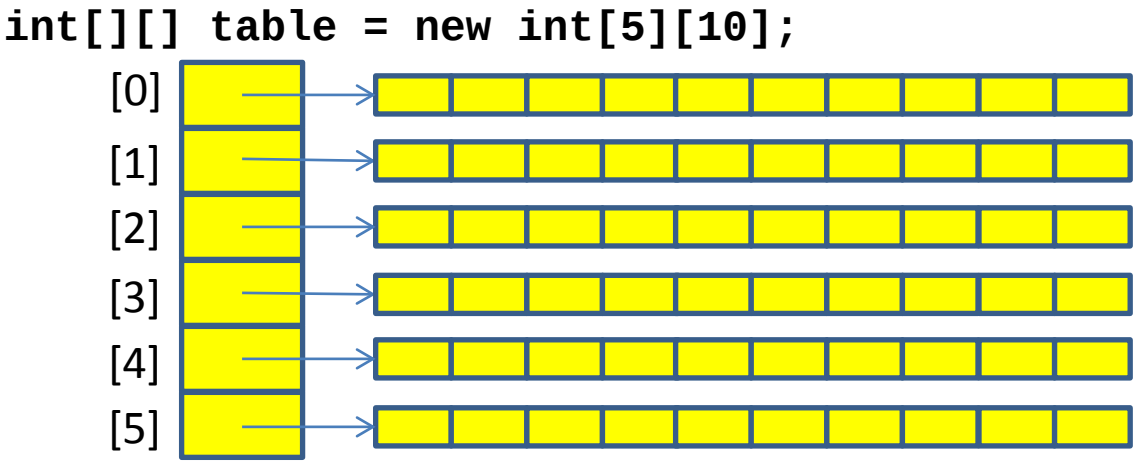


table.length är 5
table[0].length är 10

Tvådimensionella arrayer

- Fler dimensioner med fler index
- **`int [][] table = new int[5][10];`**
- table har datatypen int[][] (array of array of int)
- table[m] har datatypen int[] (array of int)
- table[m][n] har datatypen int

```
int[][] table = new int[5][10];
```

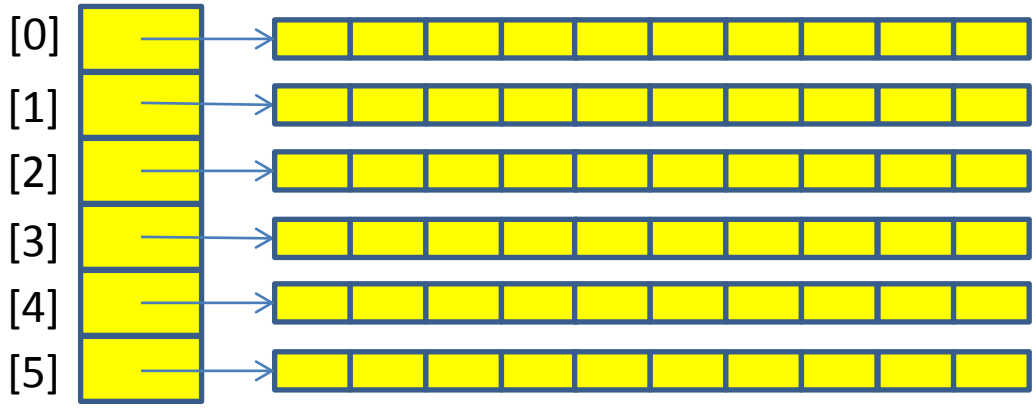


table.length är 5
table[0].length är 10

Initiering av tvådimensionella arrayer

- Görs på samma sätt: en lista per array

```
int [][] table =
  {  {1, 2, 1, 5, 7, 3, 12},
     {0, 0, 8, 3, 2, 8, 11},
     {2, 2, 6, 5, 2, 7,  9}  };
```

Tre arrayer med sju int i varje.

Multidimensionella arrayer

- Fortsätt på samma sätt:
  - `int [][][] cube = new int[3][3][3];`
- Tänk på att *allt* nödvändigt minne allokeras
- Mångdimensionellt data har ofta hål i sig

Klassen java.util.ArrayList

- Växer och krymper dynamiskt
- Endast en dimension

```
// Deklarera en referensvariabel och skapa en tom ArrayList.
ArrayList<String> nameList = new ArrayList<String>();
// Ladda in några namn
nameList.add(”Pelle”);
nameList.add(”Eva”);
nameList.add(”Lisa”);
int location = nameList.indexOf(”Pelle”); // Hitta Pelle
nameList.remove(location); // Ta bort Pelle
```

ArrayList är praktisk, men insättning och borttag i början är ineffektivt

arrayer

SLUT FÖR DENNA GÅNG