TDA 545: Objektorienterad programmering

Föreläsning 6:
Metoder och fält (arrays)

Magnus Myréen

Chalmers, läsperiod 1, 2015-2016

I (föregående och) denna föreläsning

Läsanvisning: kap 2 & 13

- meddelanden och metoder
- informationsdöljande och inkapsling
- skapa och använda färdiga objekt !
- primitiva variabler kontra objektvariabler
- 3 tester på likhet
- metoder
- fält (arrays)

Nästa föreläsning handlar om *Att skriva egna klasser*.

Metoder!

Metoder = funktioner i Java

Definition av en metod:

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

Definition av en metod:

metodhuvud, funktionshuvud, signature

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

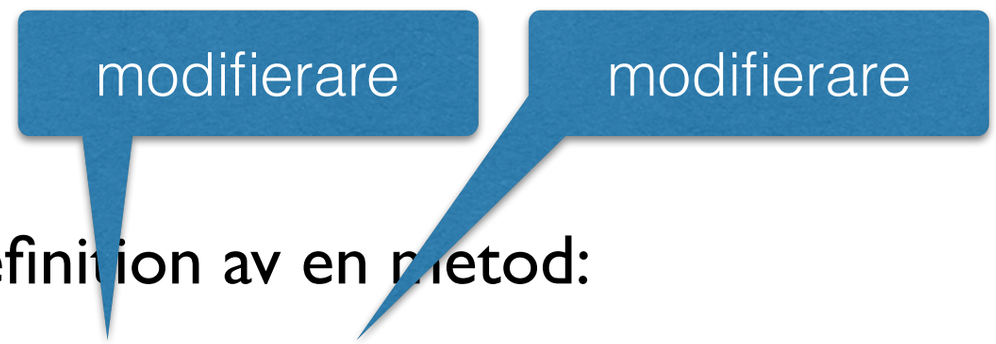
Definition av en metod:

```
public static double medel(int v1, int v2) {
    return (v1+v2)/2.0;
} // end medel
```

metodens kropp

Metoder!

Metoder = funktioner i Java



Definition av en metod:

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

Definition av en metod:

returvärdets typ

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

Definition av en metod:

metodens namn

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

Definition av en metod:

formell parametrar

formell parametrar

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

Metoder!

Metoder = funktioner i Java

Definition av en metod:

```
public static double medel(int v1, int v2) {
    return (v1+v2)/2.0;
} // end medel
```

retur värdet

Metoder!

Metoder = funktioner i Java

modifierare

modifierare

returvärdets typ

metodens namn

formell parameter

formell parameter

Definition av en metod:

```
public static double medel(int v1, int v2) {
   return (v1+v2)/2.0;
} // end medel
```

retur värdet

Metodens resultat kan vara av vilken typ som helst, även en klass.

Anrop t.ex.

```
int c = 52; int a = 10;
double medel1, medel2;
medel1 = medel(5, 10);
medel2 = medel(a, c);
```

aktuell parameter

aktuell parameter

En klass med två metoder

```
import java.util.Scanner;
public class EnkelMatematik {

    public static double medel(double v1,double v2) {
      return (v1 + v2)/2.0;
    } // end medel;

    public static void main(String[] args){
      Scanner myInput = new Scanner(System.in);
      double tal1, tal2, mv;
      System.out.println("Ange 2 tal");
      tal1 = myInput.nextDouble();
      tal2 = myInput.nextDouble();
      mv = medel(tal1, tal2);
      System.out.print("Medelvärdet är: ");
      System.out.println(mv);
    } // end main

} // end EnkelMatematik
```

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(tal1,tal2);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(tal1,tal2);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(tal1,tal2);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(25.0,tal2);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(25.0,tal2);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

de formella parametrarna är vanliga variabler för metoden …

… utgångsvärdena får de från metodanropet.

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: v1
25.0

typ: double
namn: v2
15.0

typ: double
namn: mv

typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

Evaluering av metodanropet

```
return (v1 + v2)/2.0;
```

typ: double
namn: v1

25.0

typ: double
namn: v2

15.0

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
return (v1 + v2)/2.0;
```

typ: double
namn: v1

25.0

typ: double
namn: v2

15.0

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
return (v1 + v2)/2.0;
```

typ: double
namn: v1

25.0

typ: double
namn: v2

15.0

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
return (25.0 + 15.0)/2.0;
```

typ: double
namn: v1

25.0

typ: double
namn: v2

15.0

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

typ: double
namn: tal1

25.0

typ: double
namn: tal2

15.0

Evaluering av metodanropet

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

```
return (40.0)/2.0;
```

typ: double
namn: v1
25.0

typ: double
namn: v2
15.0

typ: double
namn: mv

typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

Evaluering av metodanropet

```
return 20.0;
```

typ: double
namn: v1
25.0

typ: double
namn: v2
15.0

```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv

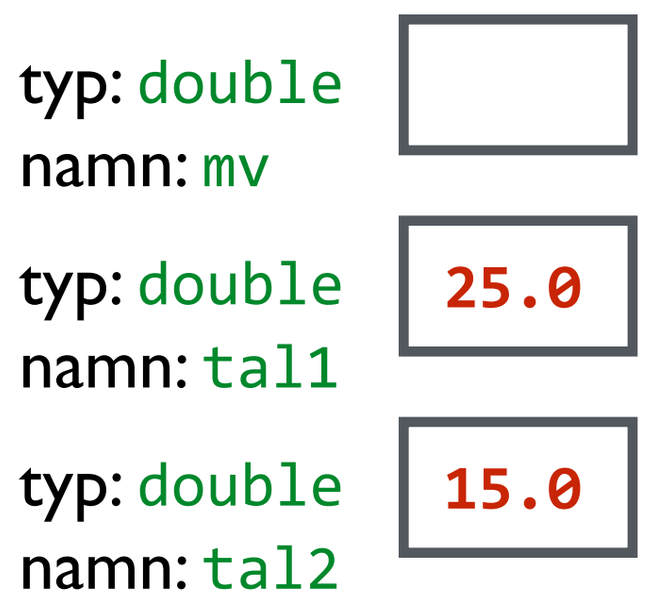
typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

Evaluering av metodanropet

```
return 20.0;
```

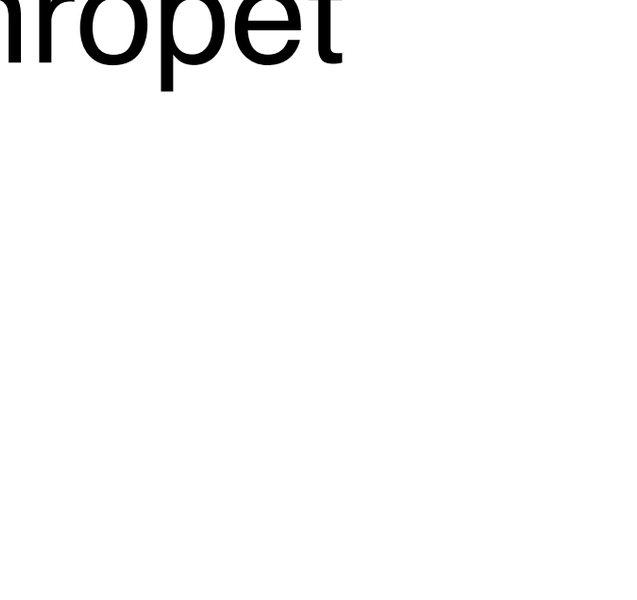
```
mv = medel(25.0,15.0);
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```



Evaluering av metodanropet

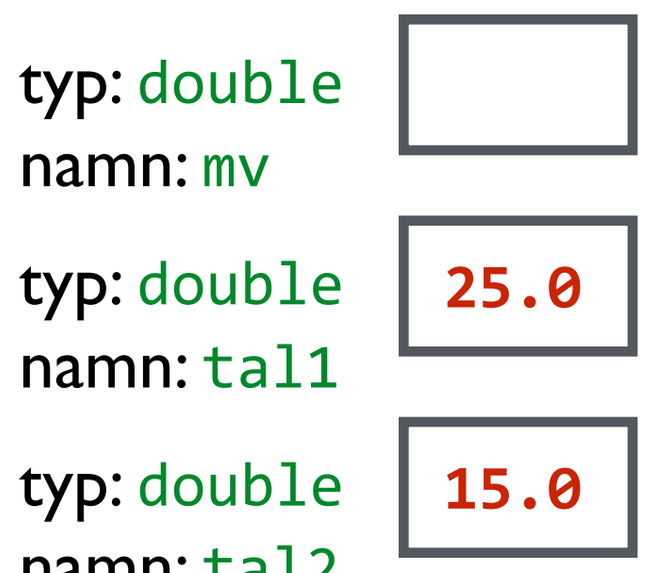
```
return 20.0;
```

```
mv = 20.0;
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```



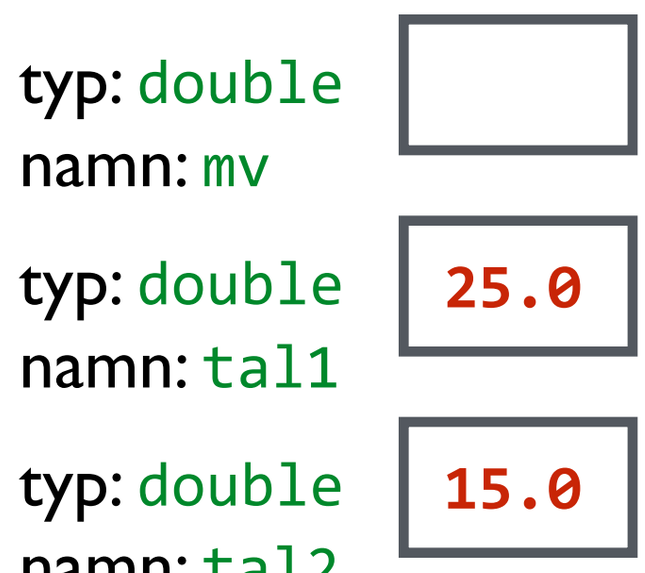
Evaluering av metodanropet

```
mv = 20.0;
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```



Evaluering av metodanropet

```
mv = 20.0;
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```



Evaluering av metodanropet

```
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: `double`
namn: `mv`
`20.0`

typ: `double`
namn: `tal1`
`25.0`

typ: `double`
namn: `tal2`
`15.0`

Evaluering av metodanropet

```
System.out.print("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: mv
20.0

typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

Evaluering av metodanropet

```
System.out.println(mv);
```

Utskrift:

```
"Medelvärdet är: "
```

typ: `double`
namn: `mv`
`20.0`

typ: `double`
namn: `tal1`
`25.0`

typ: `double`
namn: `tal2`
`15.0`

Evaluering av metodanropet

```
System.out.println(mv);
```

Utskrift:

```
"Medelvärdet är: "
```

typ: `double`
namn: `mv`
`20.0`

typ: `double`
namn: `tal1`
`25.0`

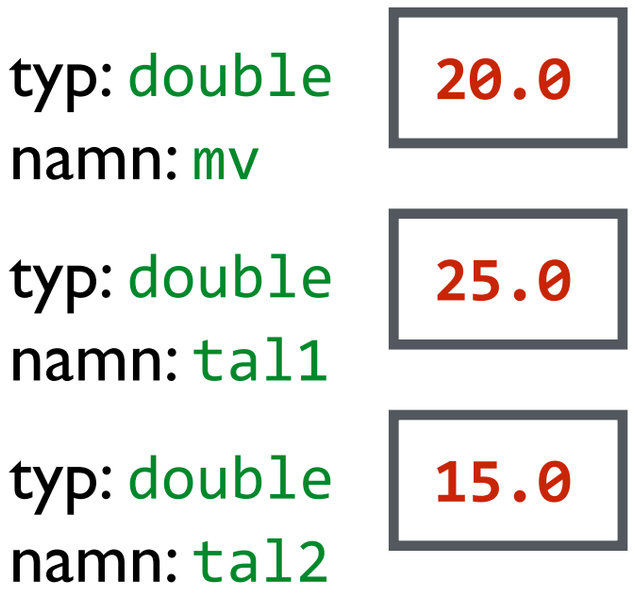
typ: `double`
namn: `tal2`
`15.0`

Evaluering av metodanropet

```
System.out.println(20.0);
```

Utskrift:

```
"Medelvärdet är: "
```



Evaluering av metodanropet

Koden har kört klart

Utskrift:

```
"Medelvärdet är: 20.0\n”
```

typ: `double`
namn: `mv`
`20.0`

typ: `double`
namn: `tal1`
`25.0`

typ: `double`
namn: `tal2`
`15.0`

Parameteröverföring

nya lokala variabler skapas

```
public static double medel(double v1,double v2) {
  return (v1 + v2)/2.0;
} // end medel;
```

de formella parametrarna är vanliga variabler för metoden …

… utgångsvärdena får de från metodanropet.

```
mv = medel(25.0,15.0);
System.out.println("Medelvärdet är: ");
System.out.println(mv);
```

typ: double
namn: v1
25.0

typ: double
namn: v2
15.0

typ: double
namn: mv

typ: double
namn: tal1
25.0

typ: double
namn: tal2
15.0

I Java överförs alltid parametrarna via värdeanrop dvs värdet av den aktuella parametern **kopieras över** till den formella parametern.

Metoder = funktioner och procedurer

"Använd en funktion om du kan, en procedur om du måste."

Funktioner *beräknar värden*. De utvidgar uttrycksdelen av språket.

Ett funktionsanrop:

- är ett uttryck
- och har en returtyp (dvs inte void)
- beräknas till ett värde som returneras.

```
mv = medel(tal1, tal2);
```

Procedurer *utför åtgärder*. De utvidgar satsdelen av språket.

Ett proceduranrop:

- är ett sats
- returnerar inget värde (returtyp = void)

```
System.out.print("Störst är: ");
```

Synbarhet (Scope)

En variabel/metod "syns" från det den deklareras tills dess att blocket den deklarerats i tar slut.

```
int global = 4;
if (<villkor>) {
  int local = 3;
  // här finns local och global
  ...
} else {
  // här finns inte local
  // men global finns
  ...
}
```

```
static double sqr(double x){
  double tmp = x*x;
  // här finns tmp och x
  return tmp;
}
// här finns inte tmp och inte x
```

Overloading (Överlagring)

Två eller flera metoder kan ha samma namn
om dom skiljer sig åt i sina parametrar.

```
public static int max(int a, int b) {
  if (a > b) {
    return a;
  } else {
    return b;
  }
}

public static int max(int a, int b, int c) {
  return max(a,max(b,c));
}

public static double max(double a, double b) {
  if (a > b) ...
}
```

Att vända på en sträng (igen!)

Uppgift:

Skriv ***en metod*** som vänder om en sträng.

Exempel: “Hello!” bör bli “!olleH”

Tips:

Gör så här:

```
str är "Hello!", result är ""
str är "Hello!", result är "H"
str är "Hello!", result är "eH"
str är "Hello!", result är "leH"
str är "Hello!", result är "lleH"
str är "Hello!", result är "olleH"
str är “Hello!", result är “!olleH"
```

Att vända på en sträng (igen!)

Lösning:

```
public static String rev(String str) {
  String result = "";
  for (int i=0; i<str.length(); i++) {
    result = str.charAt(i) + result;
  }
  return result;
}
```

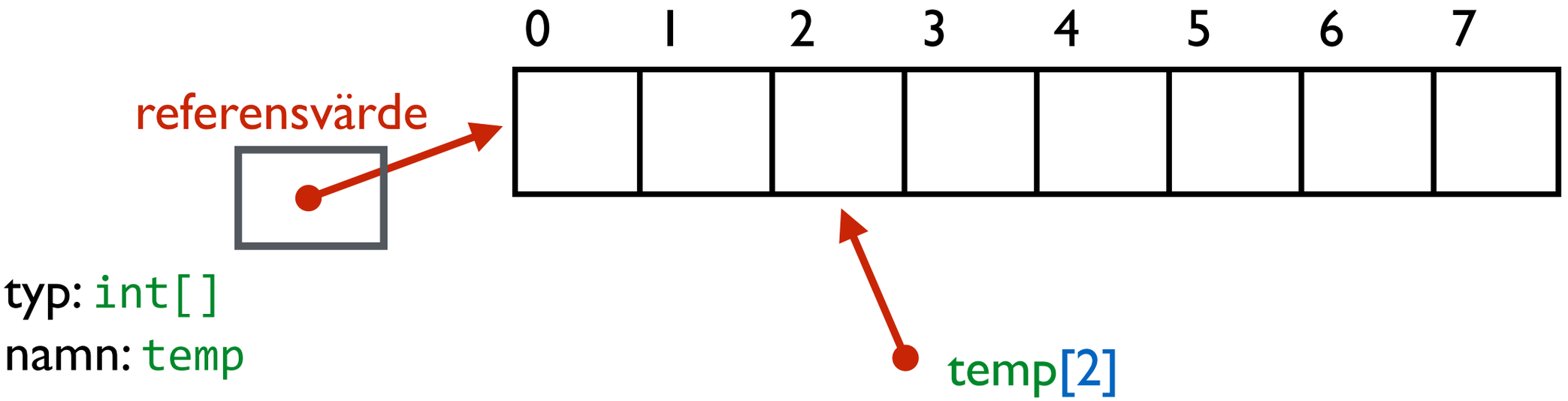
Tips:

Gör så här:

```
str är "Hello!", result är ""
str är "Hello!", result är "H"
str är "Hello!", result är "eH"
str är "Hello!", result är "leH"
str är "Hello!", result är "lleH"
str är "Hello!", result är "olleH"
str är “Hello!", result är “!olleH"
```

Fält, vektorer, matriser, arrays

ett fält = ett ‘block’ av värden



- har numrerade komponenter
- komponenterna selekteras med index av diskret typ
- indexeras från noll till length-1
- alla komponenter är av samma typ
- komponenterna kan vara objekt

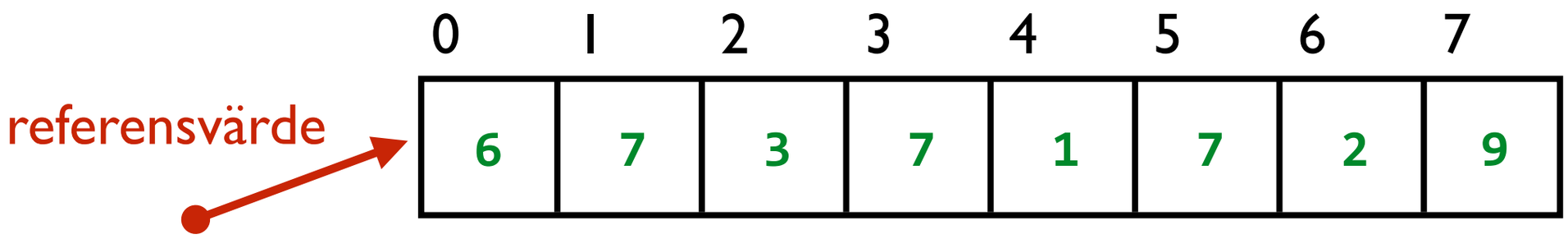
Att skapa fält

Deklarera ett fält med 8 heltal:

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

… eller deklaration och ‘snabbtilldelning’ med *array initializer*

```
int[] temp = {6,7,3,7,1,7,2,9};
```

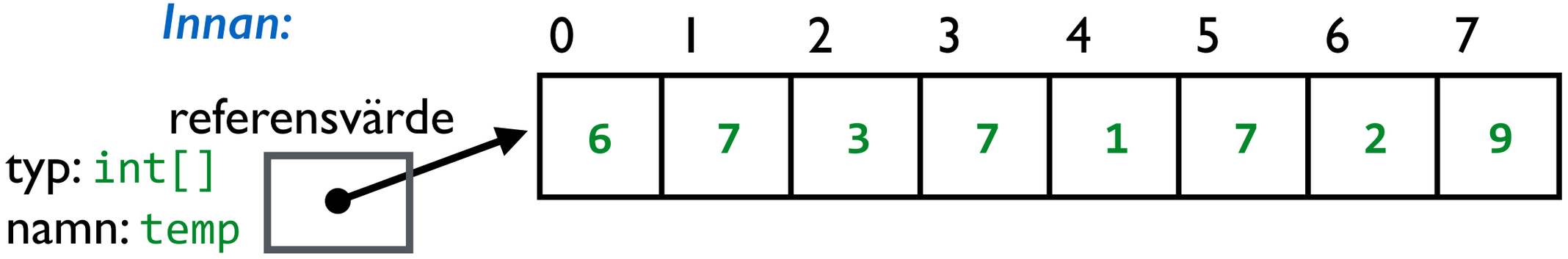


Obs. sådan tilldelning fungerar endast i deklaration, detta går ej:

```
temp = {6,7,3,7,1,7,2,9};
```

Tilldelning av fältvärden

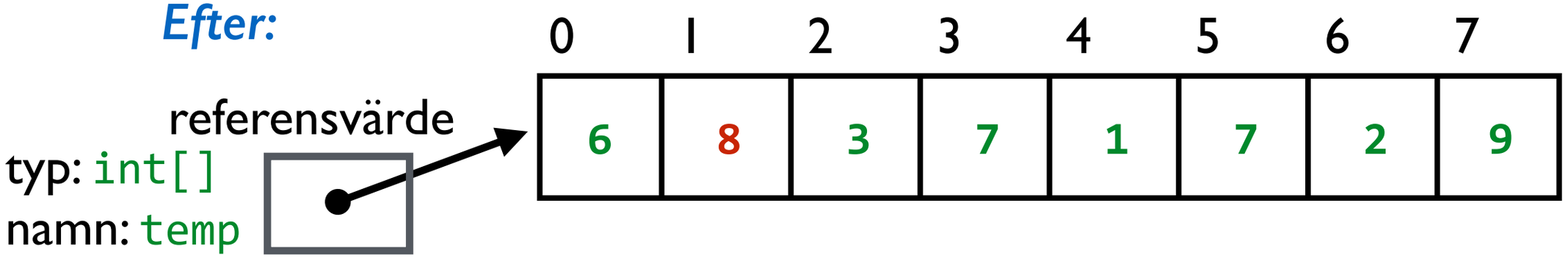
Innan:



Uppdatering av ett fältvärde:

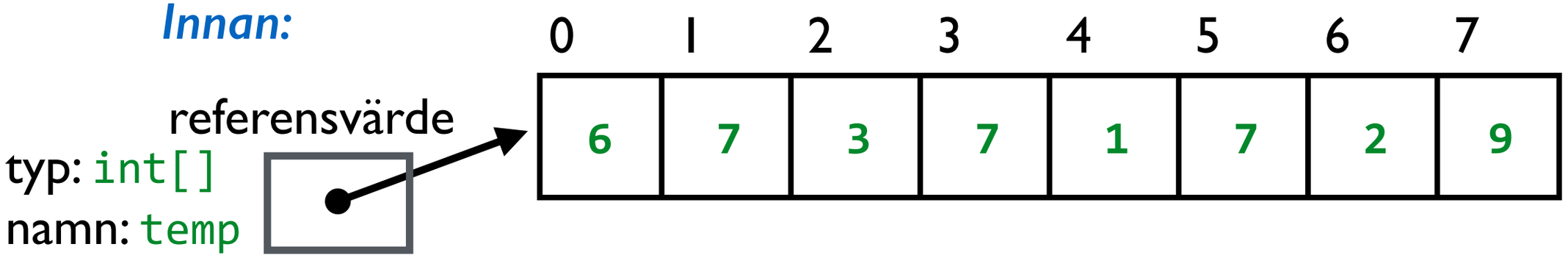
```
temp[1] = 8;
```

Efter:



Tilldelning av fältvärden

Innan:

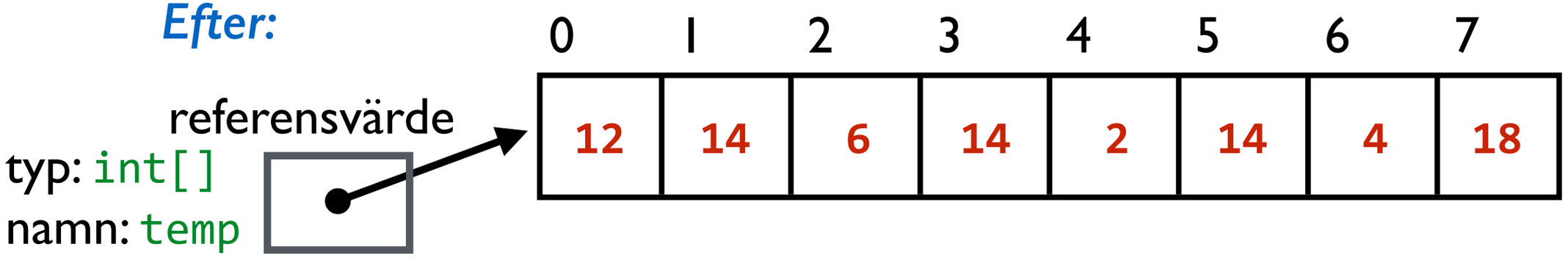


Uppdatering av alla fältvärden:

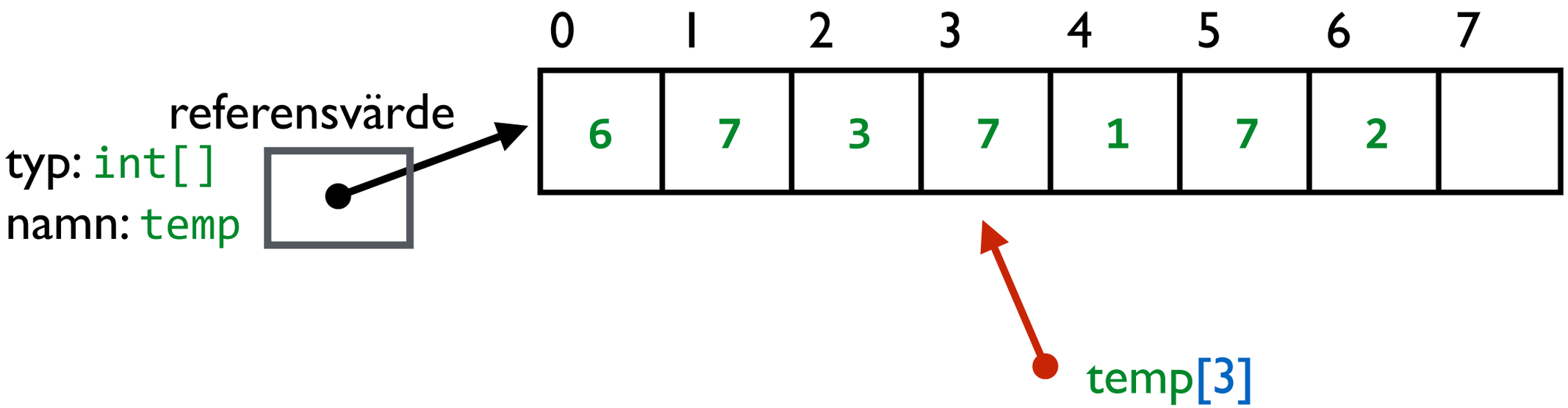
fältet längd, går ej att ändra

```
for(int i = 0; i < temp.length; i++){
  temp[i] = 2*temp[i];
}
```

Efter:



Läsning av fältvärden



Läsning av ett fältvärde:

```
int n = temp[3];
```

ändrar inte fältet

n får värdet **7**

något heltalsuttryck

Bearbeta fältets värden

Antag: `int[] temp = new temp[8];`

Mycket vanligt mönster:

```
for(int i = 0; i < temp.length; i++) {
   här: kod som bearbeta enskilda fältvärden med temp[i]
   t.ex. har tilldelning temp[i] = … temp[i] …
}
```

Vad gör denna loop?

```
for (int i = 0; i < temp.length; i++) {
    temp[i] = 1;
}
```

Den fyller fältet med ettor.

Vad gör denna loop?

```
for (int i = 0; i < temp.length; i++) {
    temp[i] = temp[i] * 2;
}
```

Den multiplicerar alla element med 2

Vad gör denna loop?

```
int antal = 0;
// i, j är vanliga namn på loop-index
for (int i = 0; i < temp.length; i++) {
    if (temp[i] < 10) {
        antal = antal + 1;
    }
}
// Här innehåller antal antalet element <10
```

Den räknar hur många element < 10

Uppgift

Summera innehållet i ett fält.

Fält och metoder

Summera innehållet i ett fält:

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

Fält och metoder

Summera innehållet i ett fält **i en metod**:

```
public static int sumArray(int[] arr) {
  int sum = 0;
  for (int i = 0; i < arr.length; i++) {
    sum = sum + arr[i];
  }
  return sum;
}
```

Fält och metoder

Skapa fält genom att anropa en metod…

returtyp: ett fält

skapar ett fält

```
public static int[] fillArray(int size) {
  int[] tmp = new int[size];
  for (int i = 0; i<tmp.length; i++) {
    tmp[i] = i; // eller nåt
  }
  return tmp;
}
```

returnerar det nya fältet

Loopar som dessa ska ni kunna skriva i sömnen…

Öva! … dvs programmera, programmera, programmera.

Kopiera ett fält

```
int[] f1 = {0,1,2,3,4,5};
int[] f2 = new int[6];
f2 = f1; // blir INTE som du tänkt
```

vad händer?

Kopiera ett fält

```
int[] f1 = {0,1,2,3,4,5};
int[] f2 = new int[6];
f2 = f1; // blir INTE som du tänkt
```

vad händer?

Man måste kopiera element för element:

```
for (int i = 0; i < f1.length; i++) {
  f2[i] = f1[i];
}
```

Ofta gör man detta i en metod:

```
static int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  for (int i=0; i<arr.length; i++){
     tmp[i] = arr[i];
  }
  return tmp;
}
```

vad händer om metoden kör tilldelning av **arr**?

hur fungerar parameter-överföringen?

… och anrop

```
f2 = copyArray(f1);
```

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray(f1);
```

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

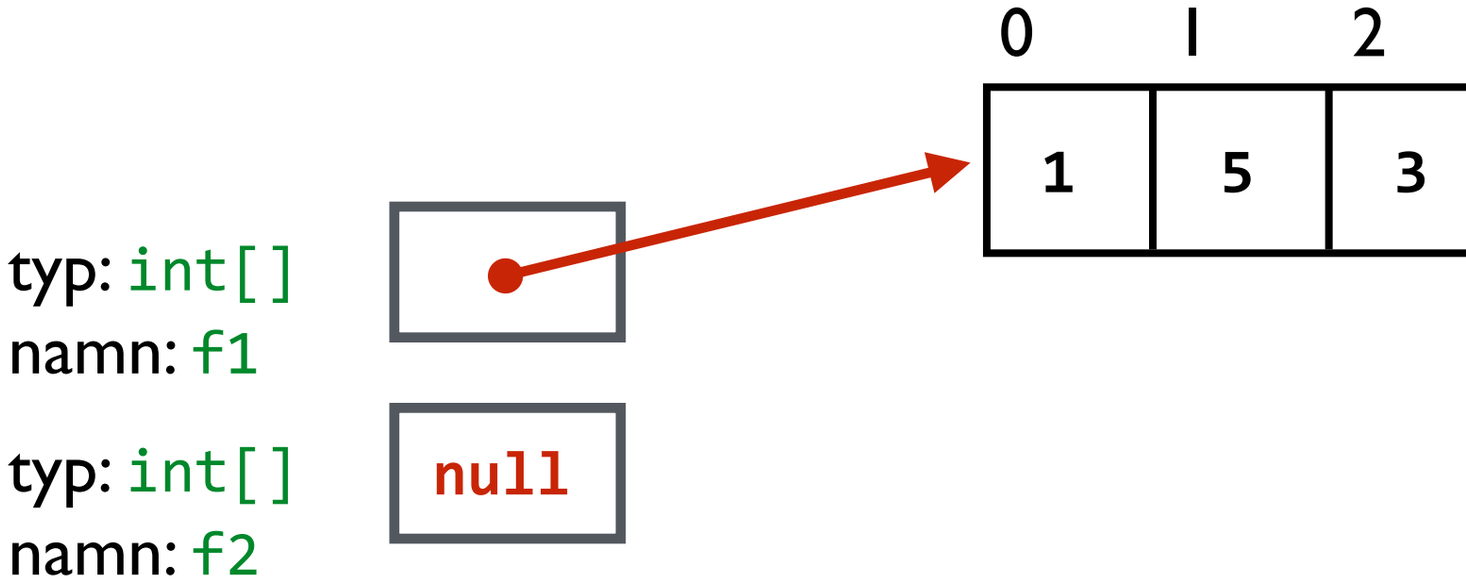
```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray(f1);
```

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray(f1);
```

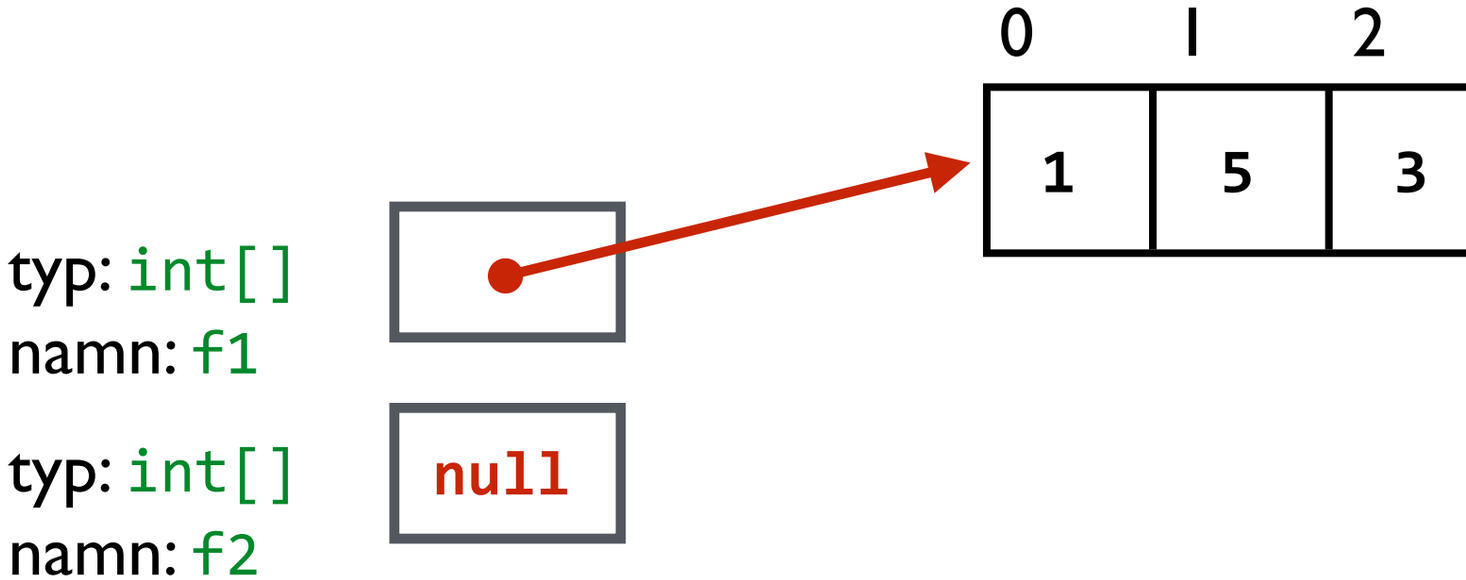


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray(f1);
```

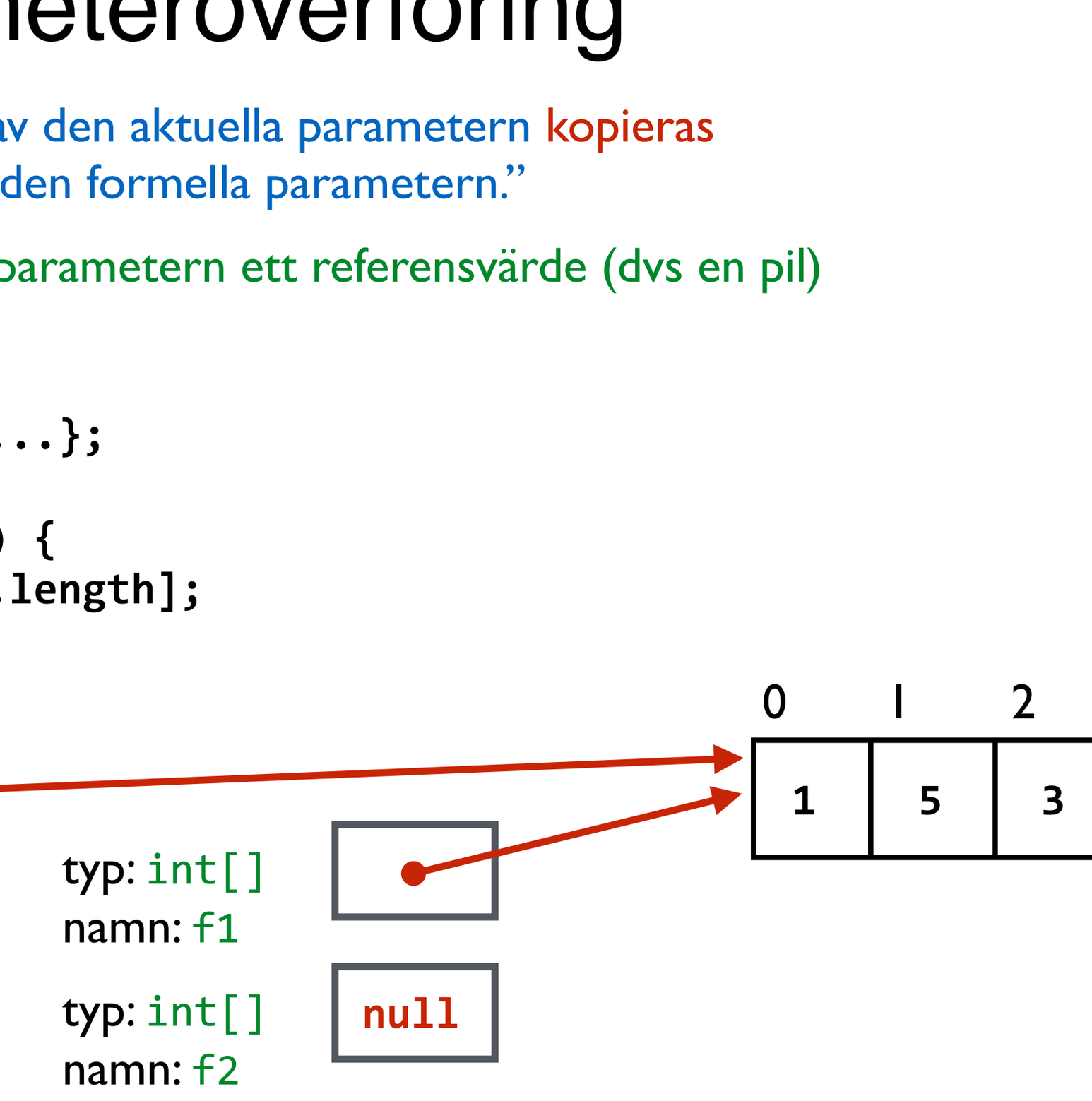


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

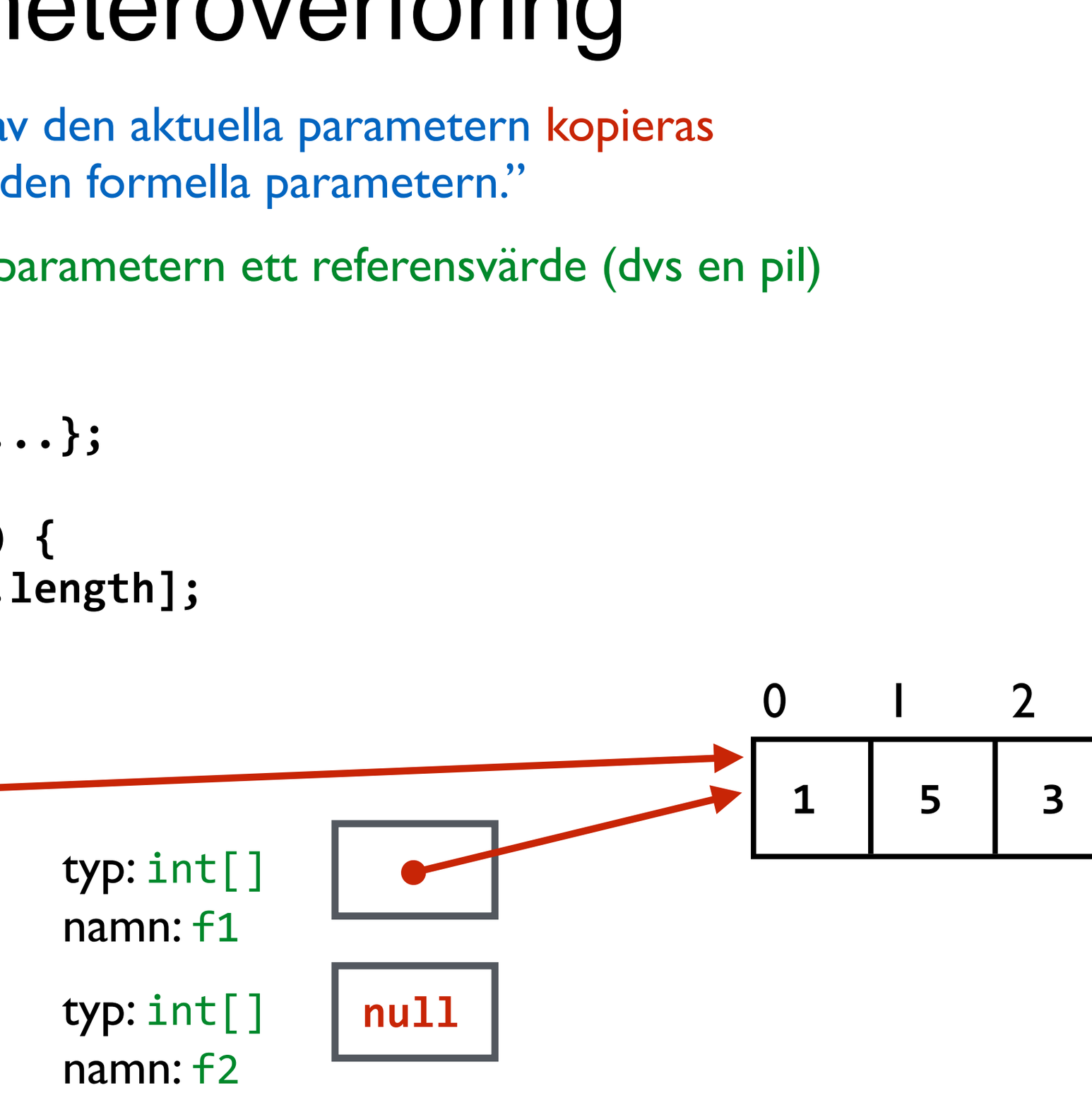


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

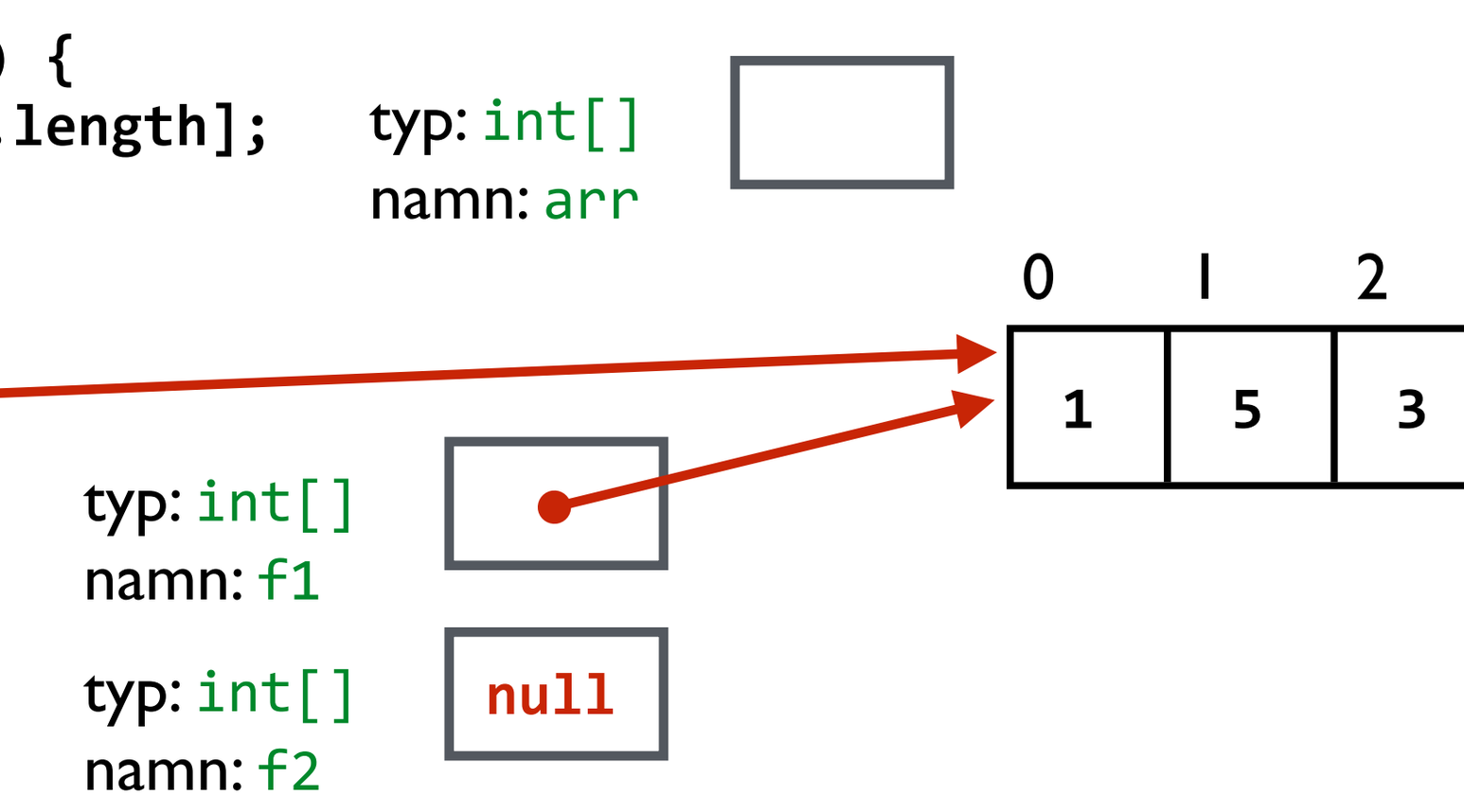


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

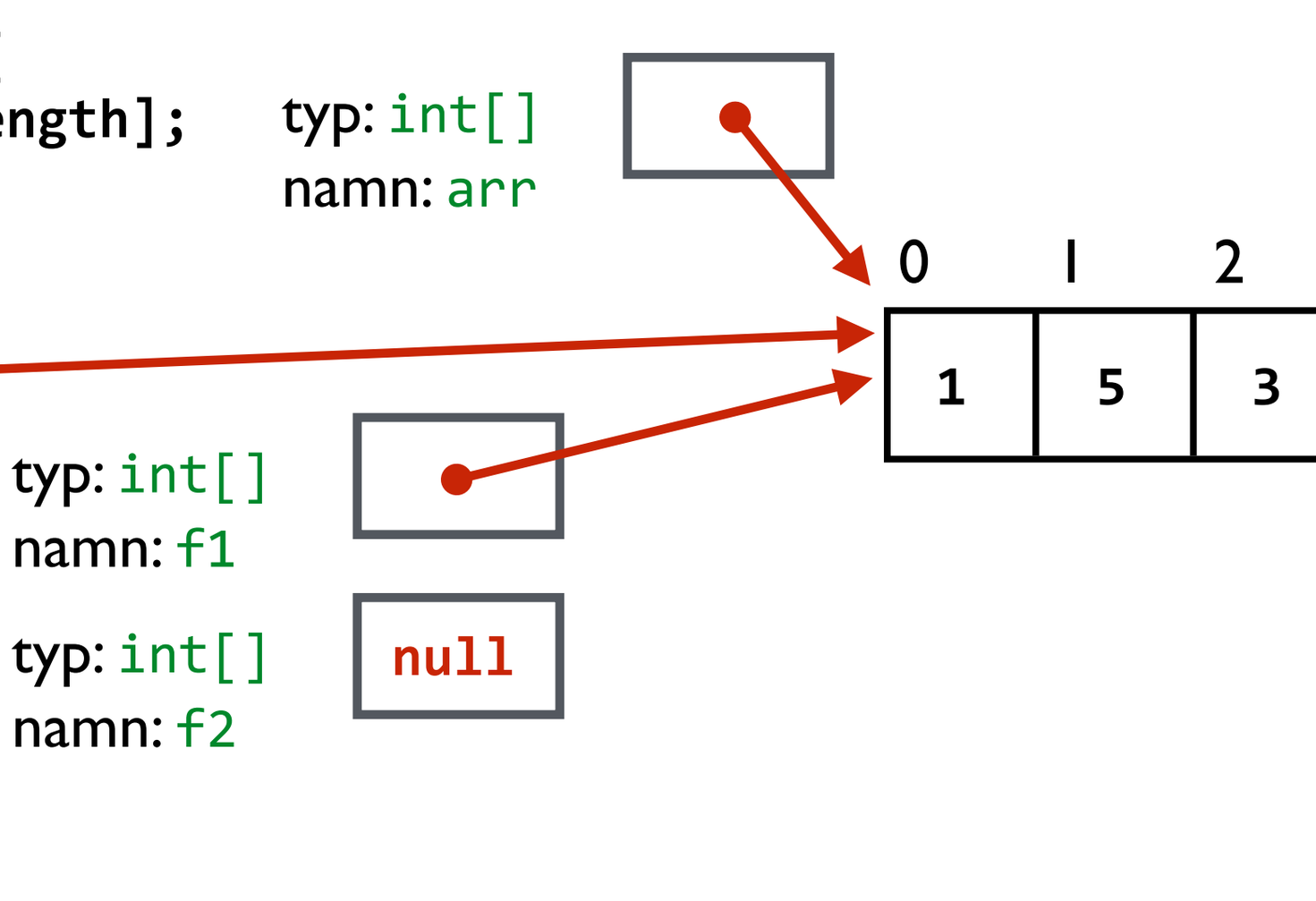


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

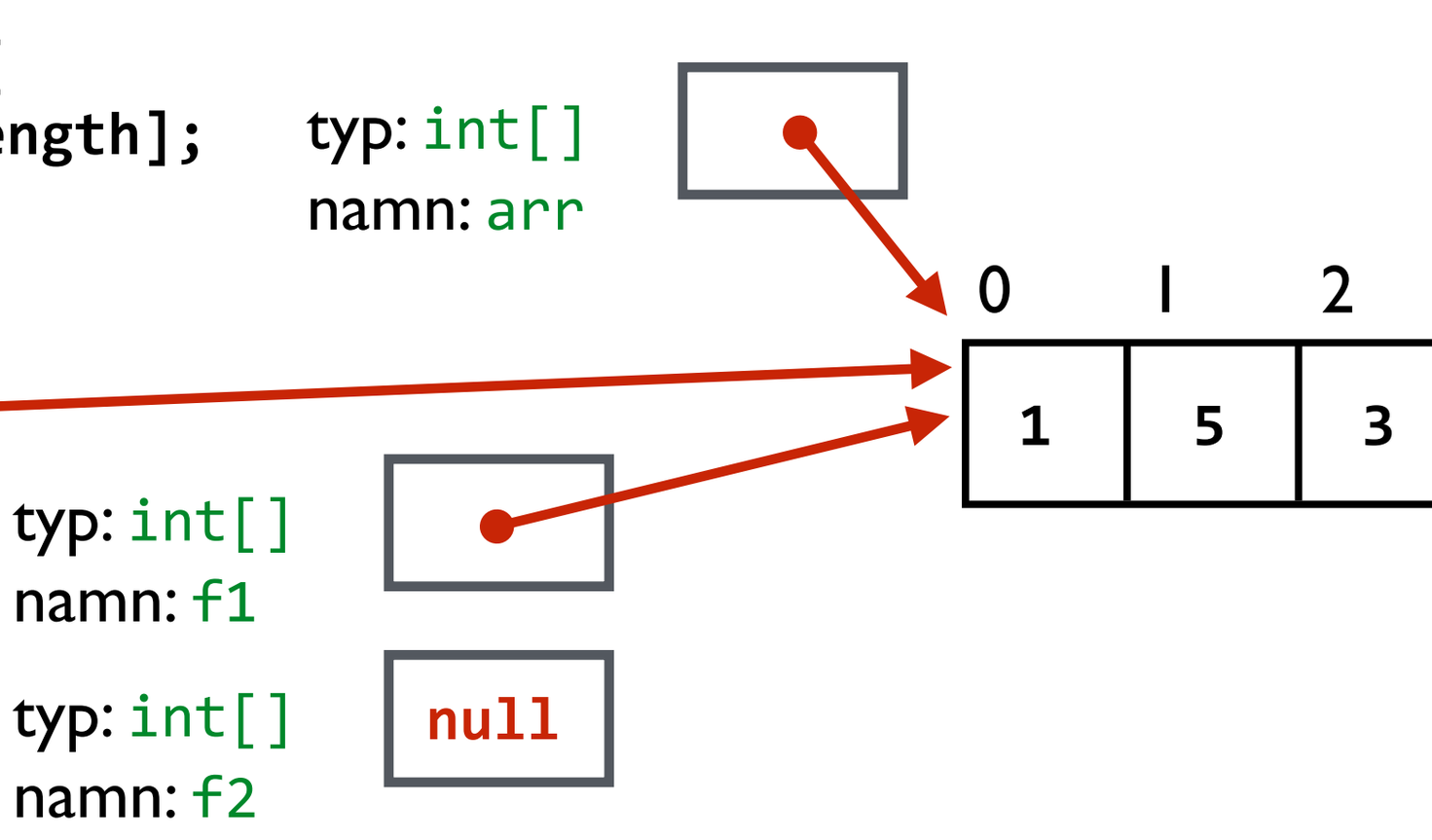


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

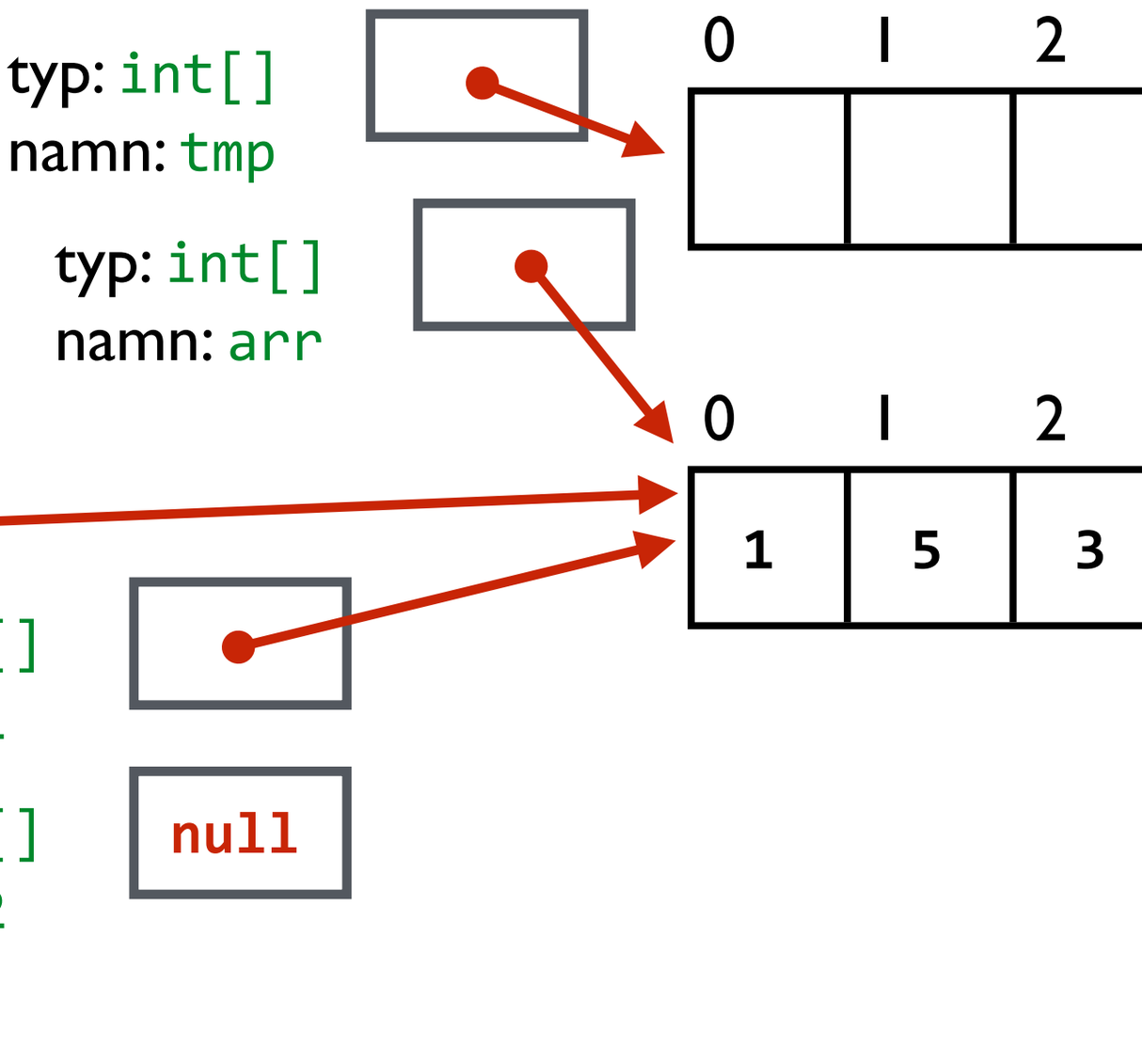


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

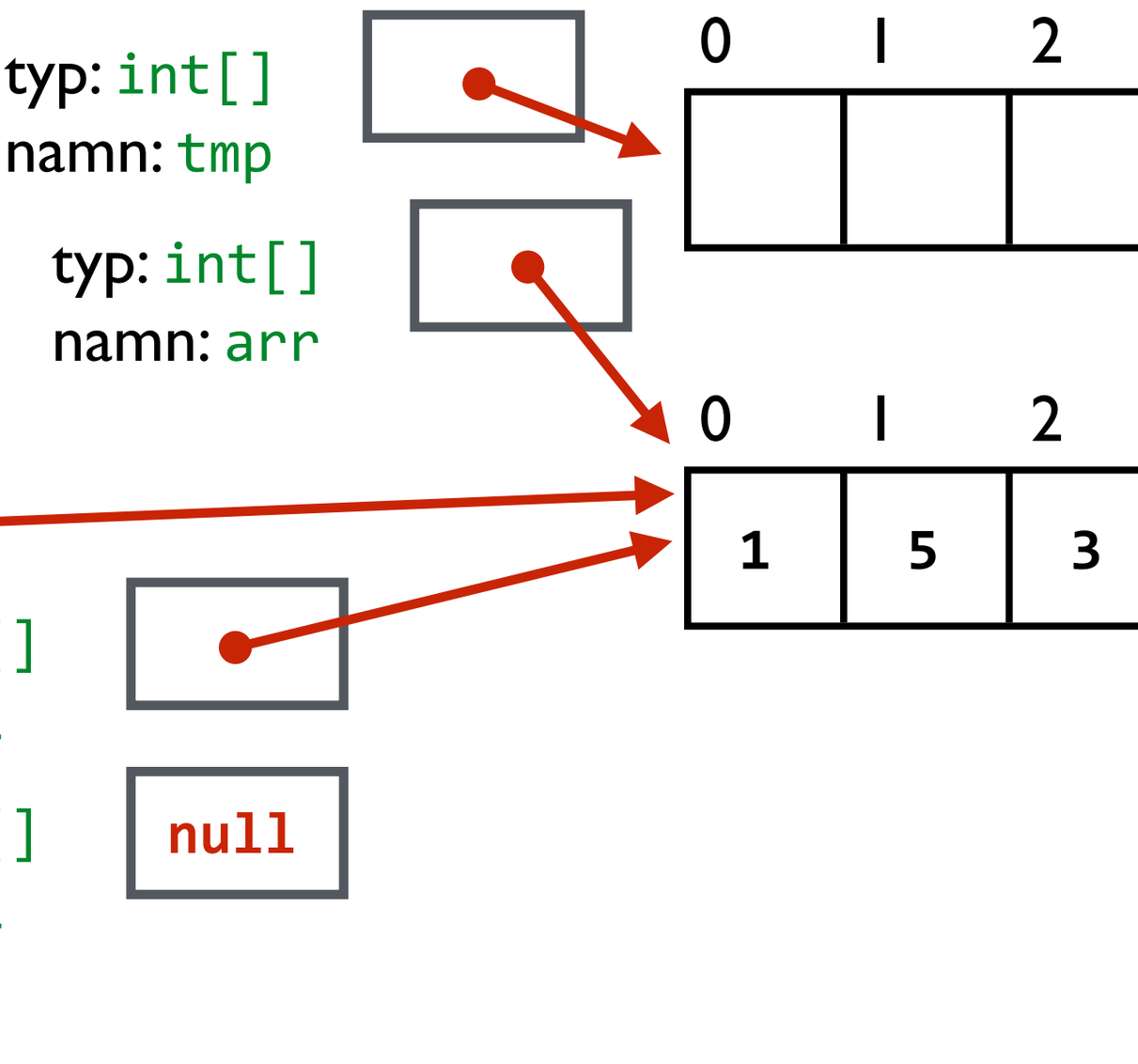


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

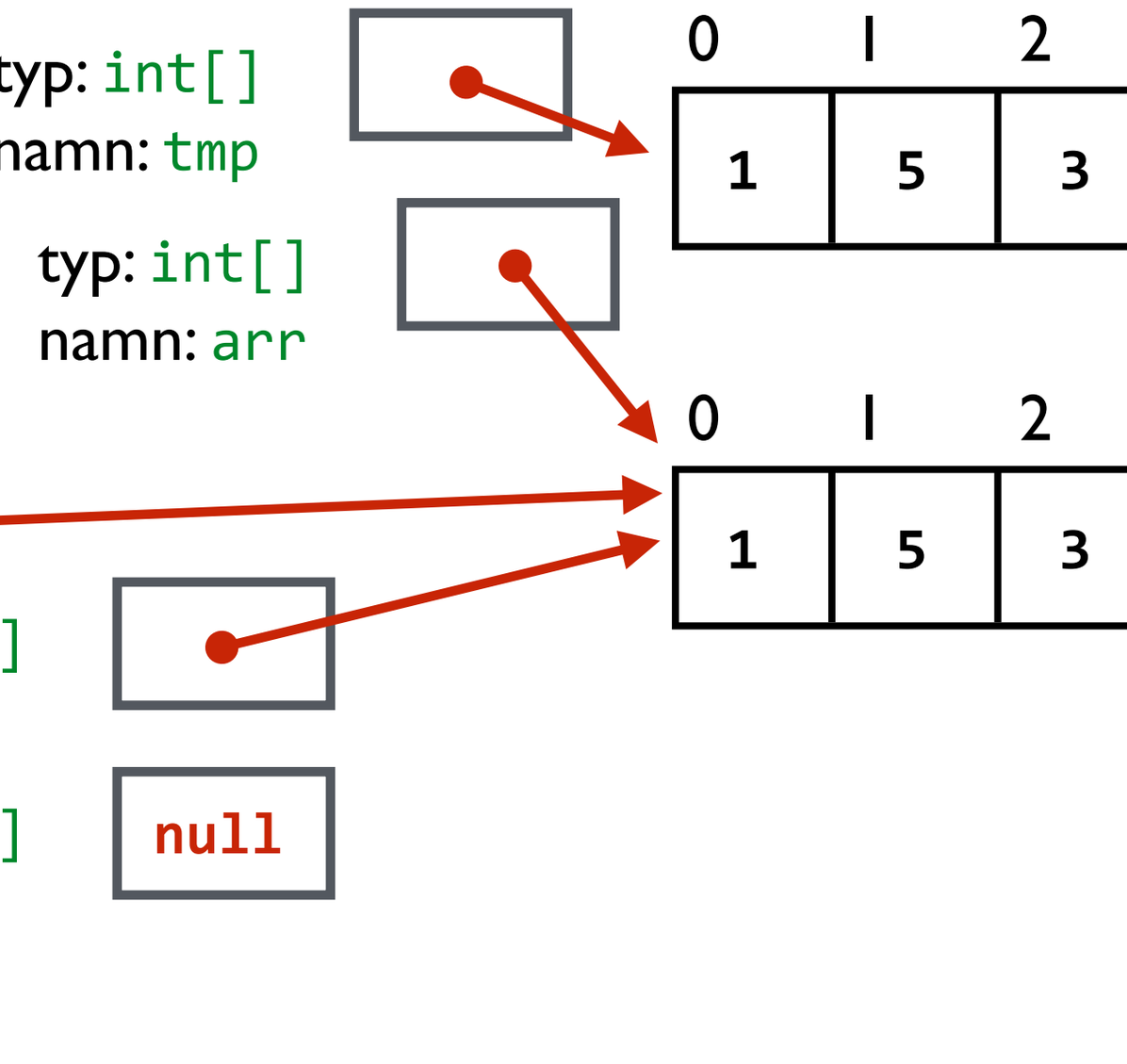


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```



Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

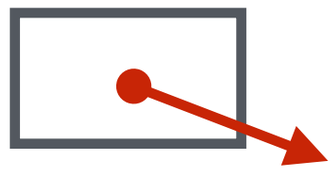
```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray( );
```

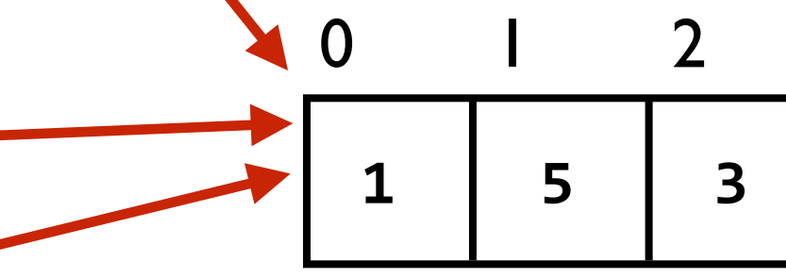
typ: int[]
namn: tmp

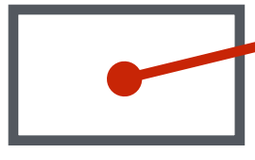
typ: int[]
namn: arr

typ: int[]
namn: f1

typ: int[]
namn: f2





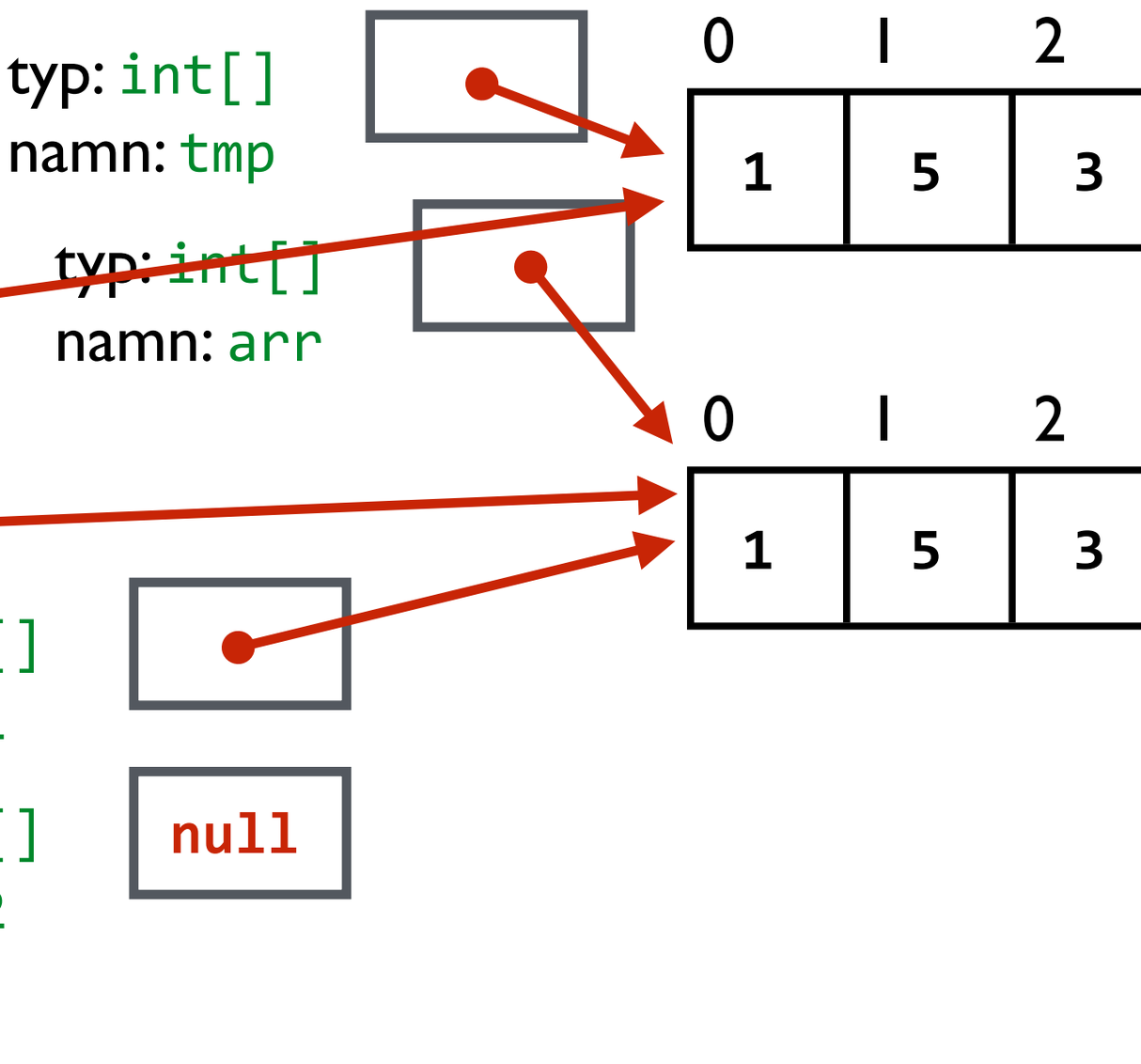


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return ;
}
f2 = copyArray( );
```

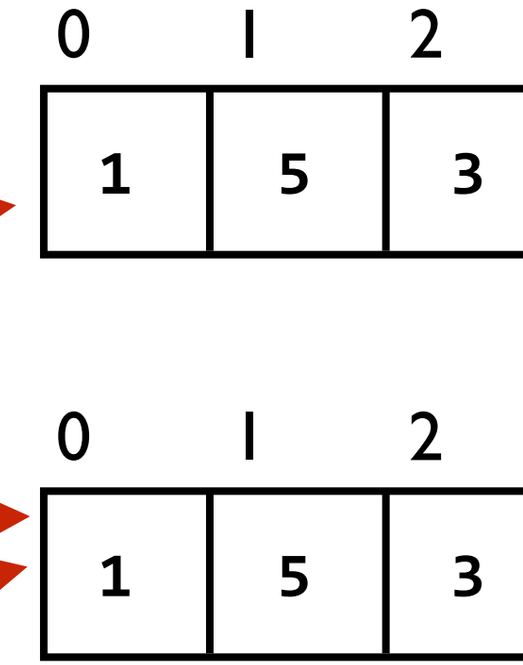


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return ;
}
f2 = copyArray( );
```



typ: `int[]`
namn: `f1`

typ: `int[]`
namn: `f2`

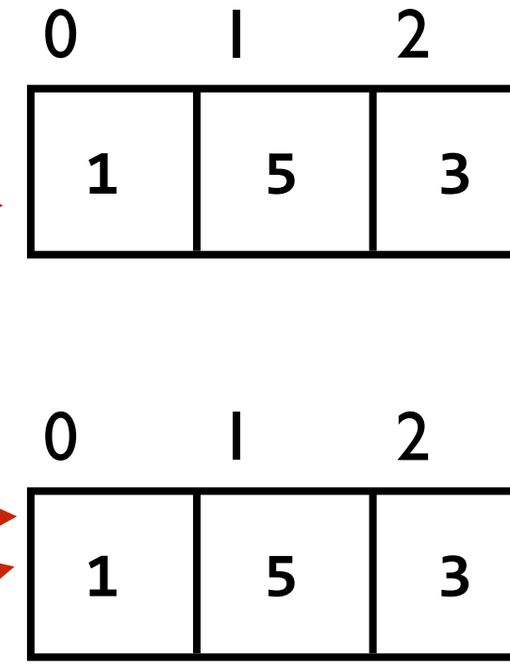
`null`

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return ;
}
f2 = copyArray( );
```



typ: `int[]`
namn: `f1`

typ: `int[]`
namn: `f2`

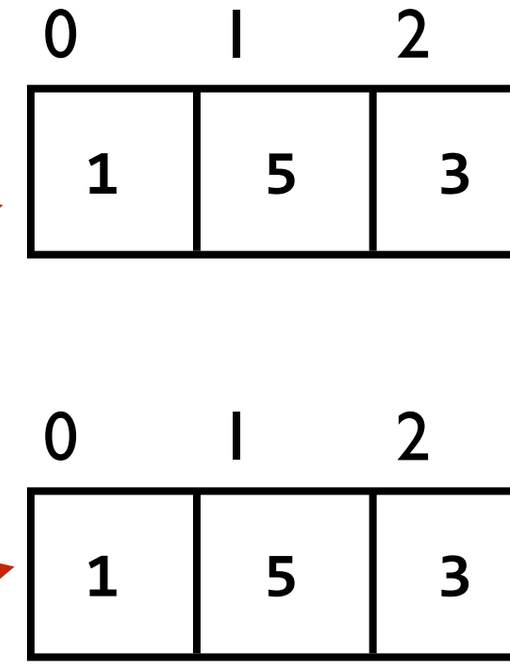
`null`

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = ;
```



typ: `int[]`
namn: `f1`

typ: `int[]`
namn: `f2`

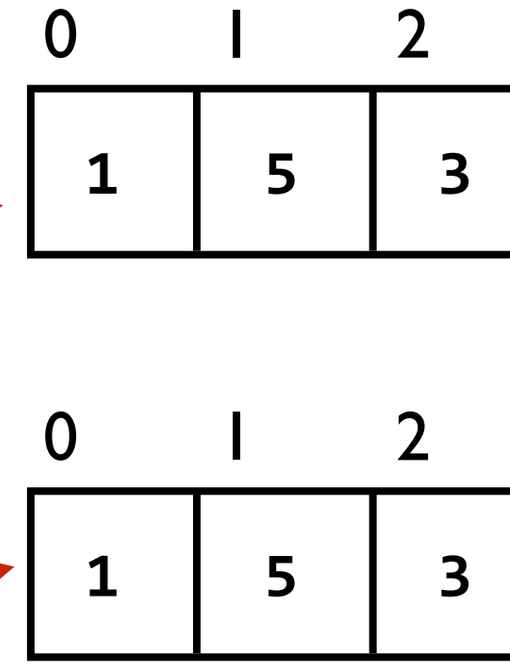
`null`

Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

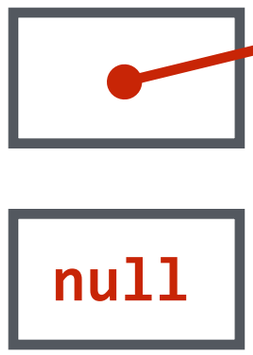
… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = ;
```



typ: `int[]`
namn: `f1`

typ: `int[]`
namn: `f2`

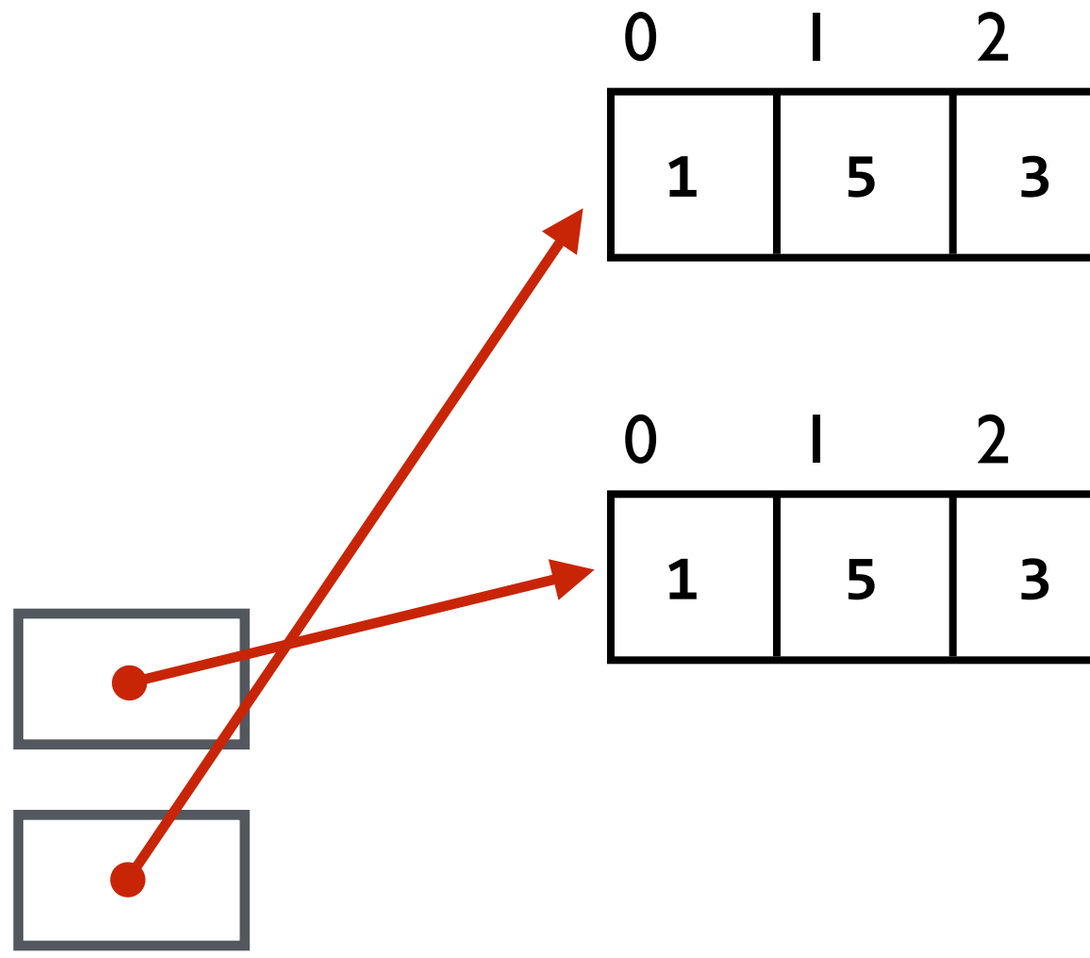


Parameteröverföring

Regeln var: "Värdet av den aktuella parametern kopieras över till den formella parametern."

… i detta fall var är parametern ett referensvärde (dvs en pil)

```
int[] f1 = {1,5,3,7,4,5,....};
int[] f2 = null;
int[] copyArray(int[] arr) {
  int[] tmp = new int[arr.length];
  ... kopiera ...
  return tmp;
}
f2 = copyArray(f1);
```



Är fält “objekt” eller primitiva?

Några skäl mot:

- annan syntax än andra objekt
- “klassen” har inget namn utan bildas genom ta namn och lägga till []
- du kan inte ärva fält
- du kan inte definiera egna metoder och lägga till klassen
- längden är en variabel, inte en funktion

Några saker som talar för att dom är det:

- språkdefinitionen säger att dom är det
- fält är referenstyper (har referensvärde)
- allokeras med “**new** …”
- allokeras på *heapen*
- förälderklassen är Object och Object klassens metoder fungerar

Så de är objekt, men lite speciella objekt.

Vanligaste felen med fält

"null pointer exception"

```
int[] f1 = {0,1,2,3,4,5};
int[] f2;
for (i = 0; i < f1.length; i++) {
  f2[i] = f1[i];
}
```

```
Test.java:10: variable f2 might not have been initialized
f2[i] = f1[i];
^
1 error
```

```
f2 = f1; // fungerar men ...
```

Vanligaste felen med fält (fort.)

“array index out of bounds”

går ett steg för långt

```
double sum = 0.0;
for (i = 0; i <= f1.length; i++) {
  sum = sum + f1[i];
}
```

Vanligaste felen med fält (fort.)

Man glömmer att kolla om indata är null…

```
static int firstElement(int[] arr) {
  return arr[0];
}
```

Bättre:

```
static int firstElement(int[] arr) {
  if (arr == null) {
    throw new IllegalArgumentException("empty array");
  } else {
    return arr[0];
  }
}
```

Vanligaste felen med fält (fort.)

Man glömmer att kolla om indata är null…

```
static int firstElement(int[] arr) {
  return arr[0];
}
```

Ännu bättre:

```
static int firstElement(int[] arr) {
  if (arr == null) {
    throw new IllegalArgumentException("empty array");
  } else if (arr.length < 1) {
    throw new IllegalArgumentException("empty array");
  } else {
    return arr[0];
  }
}
```

Flerdimensionella fält (matriser)

Är fält med fält som värden.
Raderna behöver inte vara lika långa.

```
int[][] m = new int[3][6];

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

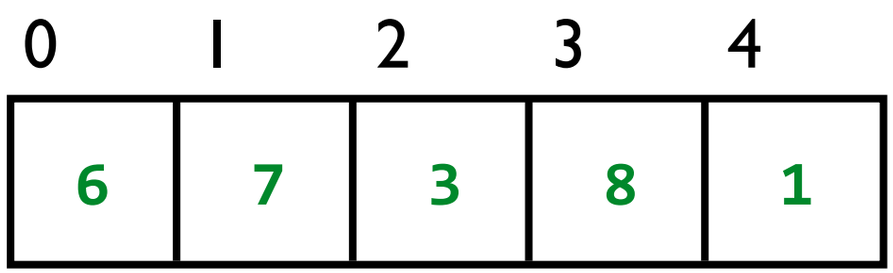
```
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
int[][] m2 = {
               {1, 2, 3, 4, 5, 6},
               {1, 1, 1, 1},
               {6, 5, 4, 3, 2, 1}
             };
```

Uppgift

Skriv kod som vänder om alla element i ett fält (dvs array).

Före:



Efter: