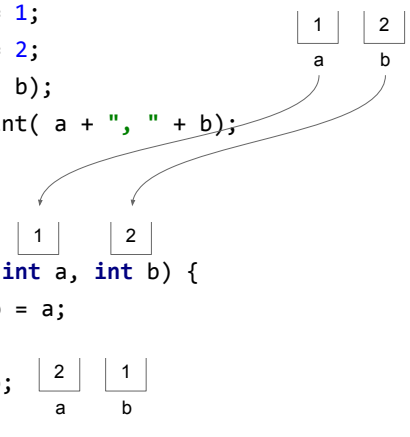
ex3methods

Joachim von Hacht

Variabler och Åtkomst

```
void program() {
   int a = 1;
   int b = 2;
   swap(a, b);
   out.print( a + ", " + b);
}

void swap(int a, int b) {
   int tmp = a;
   a = b;
   b = tmp;
}
```



2

En metod kan aldrig komma åt lokala variabler i en annan metod

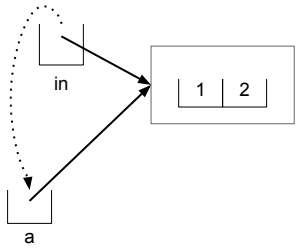
- Antag att vi vill skriva en metod, swap, som byter plats på argumenten!
- När vi anropar metoden swap kopieras värdena i a och b till parametrarna (lokala variabler), därefter byter dessa värden
- … inget händer med a och b i program(),… swap kan aldrig komma åt a och b.
- Att variabler och parametrar råkar heta samma sak här göra att man tror att man ändrar variabler utanför metoden (vilket alltså inte går)

Värden kommer metoder åt genom att skicka dessa mellan sig (parametrar/returvärden).
OBS! Däremot kan metoder komma åt objekt “utanför” … om parametrarna är referenser. Mer strax.

Array-parametrar

```
void program() {
   int[] in = { 1,2 };
   zero(in);
    // in is now [ 0, 0 ]
}


void zero( int[] a ){
   for(int i = 0; i < a.length ; i++){
       a[i] = 0;
    }
}
```



Array-parametrar är referensvariabler.

- Om en metod har en array-parametrar sker precis samma sak som vanligt vid anropet men… argumentet som kopieras är en referens!
- Parametern kommer därför att peka på samma objekt som argumentet!
- Metoden kan ändra på ett objekt utanför sig självt (variabeln "in" kan vi däremot aldrig ändra)

I koden ovan kommer array:en som variabeln in refererar att vara förändrad efter metodanropet

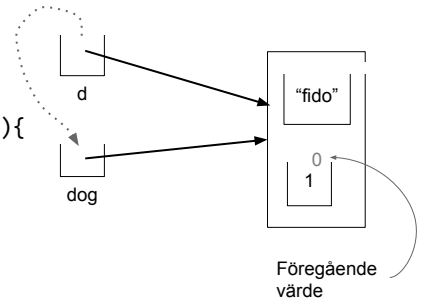
- Kallas ibland **utparameter**
- Innebär en viss risk eftersom det kan vara svårt att inse att en metod har ändrat i objektet (eftersom den är void). Undvik.

Objekt som Parameter

```
Dog d = new Dog();
incAge(d);

void incAge( Dog dog ){
   dog.age++;
}

class Dog {
   String name;
   int age;    // 0 default
}
```



Objektparametrar är referensvariabler.

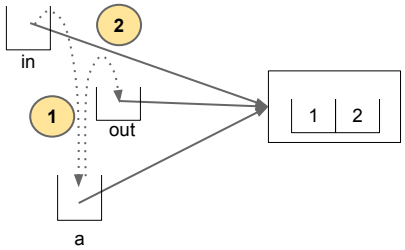
- Eftersom referensen till objektet kopiera till metodens parameter kommer den åt objektet (utanför metoden)
- På samma sätt som för en array

Array som Returtyp (1)

```
void program() {
   int[] in = { 1,2 };
   int[] out = zero(in);

}

int[] zero( int[] a ){
   for(int i = 0; i < a.length ; i++){
       a[i] = 0;
    }
    return a;  // Return parameter!
}
```



Om man returnerar en array sker samma sak som vid ett vanligt metodanrop

- Dock är returvärdet en referens

Här returnerar vi samma referens som vi skickar in.

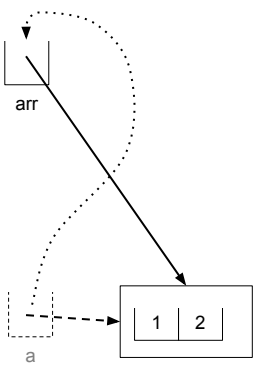
- Kan i vissa fall ge smidigare kod.

Array som Returtyp (2)

```
void program() {
   int[] arr = getArray();

}


int[] getArray(){
    int[] a = { 1, 2 };
    return a;
}  // a gone here
```



Vi kan skapa en array i en metod och skicka som returvärde.

- Den lokala variabeln a försvinner! …
- … men inte array-objektet
- För att senare kunna komma åt array-objektet måste vi spara den returnerade referensen

Objekt Returtyp

```
// Return reference to modified original object
Pair incPair(Pair pair) {
   pair.a++;
   pair.b++;
   return pair;
}

// Return reference to new object
Pair incPair2(Pair pair) {
   Pair p = new Pair(pair.a, pair.b);
   p.a++;
   p.b++;
   return p;
}
```

```
class Pair {
   int a;
   int b;

   public Pair(int a, int b) {
       this.a = a;
       this.b = b;
   }
}
```


Samma möjligheter som för arrayer.

Objekt-Arrayer som Parameter

```
Dog findOldest(Dog[] dogs){
   int index = 0;
   int maxAge = dogs[index].age;
   for( int i = 0 ; i < dogs.length ; i++){
       if( dogs[i].age > maxAge){
           index = i;
           maxAge = dogs[i].age;
       }
   }
   return dogs[index];
}
```


Kan skicka komplexa argument

- Här en parameter för en array med Dog-objekt
- Samma alternativ som för arrayer/objekt parametrar

Objekt-Array som Returtyp

```
Dog[] getDogs(){
    Dog[] dogs = { new Dog(), new Dog() };
    dog[0].name = sc.nextLine();
    dog[1].name = sc.nextLine();
    return dogs;
}
```


Komplex returtyp

- Här en array med Dog-objekt som returvärde.
- Samma alternativ som för arrayer/objekt parametrar

Sträng som Returtyp

```
final Scanner sc = new Scanner(in);

// Get user name (NOTE: No variable used)
String getName() {
   out.print("Please enter your name > ");
   return sc.nextLine();  // Return result at once
}
```


Strängar hanteras som referenser. Vi returnerar en referens.

- Här i kombination med IO.