

6. Metódy



Metóda je v podstate funkcia alebo procedúra, ktorá je definovaná pre dátový typ.

Dátový typ s atribútmi a metódami voláme **Trieda**.

(Dátové typy, ktoré sme doteraz používali boli triedy bez metód. Ako uvidíme neskôr, programy, ktoré sme doteraz v Jave používali, boli triedy bez atribútov.)

6. Metódy



Objekt „vie“ sám najlepšie, ktoré hodnoty smú a ktoré nesmú jeho atribúty nadobúdať

preto

by hodnoty atribútov objektov nemali byť priamo menené, ale na ich zmenu by sa mali použiť metódy .

7. Konštruktor

– špeciálna metóda

Ďalší princíp:

Objekt by vždy mal byť vždy v „zmysluplnom“ resp. „konzistentnom“ stave.

Pokiaľ sa nevykonáva žiadna jeho metóda.

Príklad

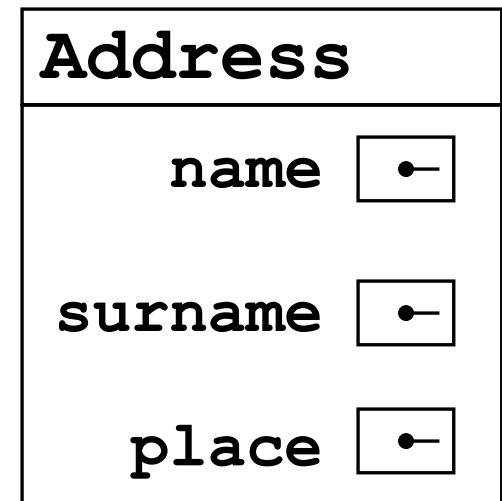
```
class Address
{ String name;
  String surname;
  String place;
}
```

```
Address adr;
```

```
adr = new Address();
```

```
// Objekt nie je žiadna „zmysluplná“ adresa
```

adr



Riešenie:



Jedna alebo viac špeciálnych metód, ktorá je volaná pri vzniku objektu sa nazýva:

Konštruktor

Príklad

```
class Address
{ private String name;
  private String surname;
  private String place;
```

Definícia konštruktora pre triedu `Address`. Tu je konštruktorom metóda s tromi parametrami.

```
Address(String n, String s, String p)
```

```
{ name = n;
  surname = s;
  place = p; }
```

```
...
```

```
}
```

```
...
```

```
Address adr = new Address("Egon", "Maier", "Berlin");
```

Volanie konštruktora! Pri volaní konštruktora budú atribúty objektu „zmysluplne“ nastavené.

Konštruktor



Konštruktor je špeciálna metóda,

- ktorá má rovnaké meno ako trieda
- Pre ktorú nie je definovaná žiadna návratová hodnota (ani `void`) a

Konštruktor iba inicializuje objekt.

- Ktorá je volaná tak, že pred ňou je kľúčové slovo `new`!

Poznámka



- V triede môže byť definovaných viacero konštruktorov, ktoré sa líšia počtom a typom parametrov.
Ktorý konštruktor sa zavolá (po **new**), závisí od typu a počtu parametrov.

Poznámka



- Trieda `Prıklad`, v ktorej nie je definovaný **žiadny** konštruktor, má automaticky **štandardný konštruktor**

```
Prıklad()  
{ }
```

Preto sme mohli doteraz vytvárať nové objekty v triedach bez toho, aby sme v týchto triedach definovali konštruktory:

```
Prıklad pr = new Prıklad();
```

totiž zavolať štandardný konštruktor.

Poznámka



□ Pozor:

Ak definujeme vlastný konštruktor, potom nebude automaticky k dispozícii štandardný konštruktor!

V takom prípade, ak potrebujeme konštruktor bez parametrov, musíme ho sami definovať!!!

Príklad

```
class Address
{ private String name;
  private String surname;
  private String place;
```

```
  Address(String n, String s, String p)
  { name = n;
    surname = s;
    place = p; } }
```

```
Address adr = new Address(); // Chyba!!!
```

```
>javac Address_book.java
```

```
Address%book.java:71: No
constructor matching Address()
```

```
found in class Address.
```

```
    Address adr = new Address();
```

```
                ^
```

```
1 error
```

Konštruktory - zhrnutie



- Každá trieda má minimálne jeden konštruktor
- Iba v jednoduchých prípadoch sa spoliehame na štandardný konštruktor
- Konštruktor v triede umožňuje „zmysluplnú“ inicializáciu novo vytvoreného objektu triedy

8. `this` – ukazovateľ sám na seba



- V metóde môžu byť použité atribúty objektu, pre ktorú bola metóda zavolaná
- V mnohých prípadoch je potrebné poznať identitu, resp. adresu samotného objektu, pre ktorý sa metóda volá
- K adrese objektu, pre ktorý sa metóda volá pristúpime pomocou kľúčového slova: `this`

this – ako špeciálny atribút



- V nejakej triede `Priklad` sa `this` správa ako keby sme definovali atribút `private Priklad this;`
- Hodnota „atribútu“ `this` však nemôže byť zmenená
- Kľúčové slovo `this` nemôže byť použité ako meno premennej alebo atribútu

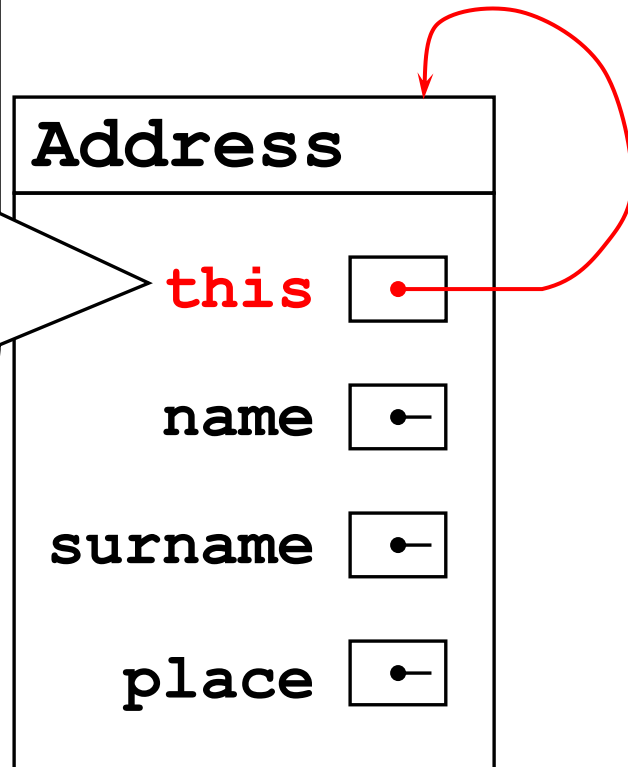
Príklad

```
class Address
{ String name;
  String surname;
  String place;
}
```

„Atribút“, ktorého hodnota nemôže byť zmenená a zároveň ktorého hodnota obsahuje adresu objektu.

adr

```
Address adr = new Address();
```



Príklad: Problém

```
class Address
{ private String name;
  private String surname;
  private String place;
```

```
Address(String name, String surname, String place)
```

```
{ name = name;
  surname = surname;
  place = place; }
```

```
...
}
```

Pozor:
Tu **nebudú** atribúty name,
surname a place objektu
zmenené!

Riešenie:

použit' this



```
class Address
{ private String name;
  private String surname;
  private String place;

  Address(String name, String surname, String place)
  { this.name = name;
    this.surname = surname;
    this.place = place; }
  ...
}
```

9. Atribúrt a metódy triedy

(`static`)



- S použitím kľúčového slova `static` definujeme premenné a metódy, ktoré sa nevutvárajú pre každý objekt zvlášť, ale existujú iba v jednej kópii pre celú triedu

Základný rozdiel

- Atribút (premenná) alebo metóda definovaná bez kľúčového slova `static` definuje pre každú inštanciu – každý objekt príslušný atribút
- Pri použití kľúčového slova `static` bude existovať atribút iba raz a v prípade ak vytvoríme objekty triedy, potom ho všetky objekty triedy spoločne zdieľajú.

Atribút definovaný použitím kľúčového slova `static` môžeme použiť aj vtedy, ak nie je vytvorený žiadny objekt danej triedy

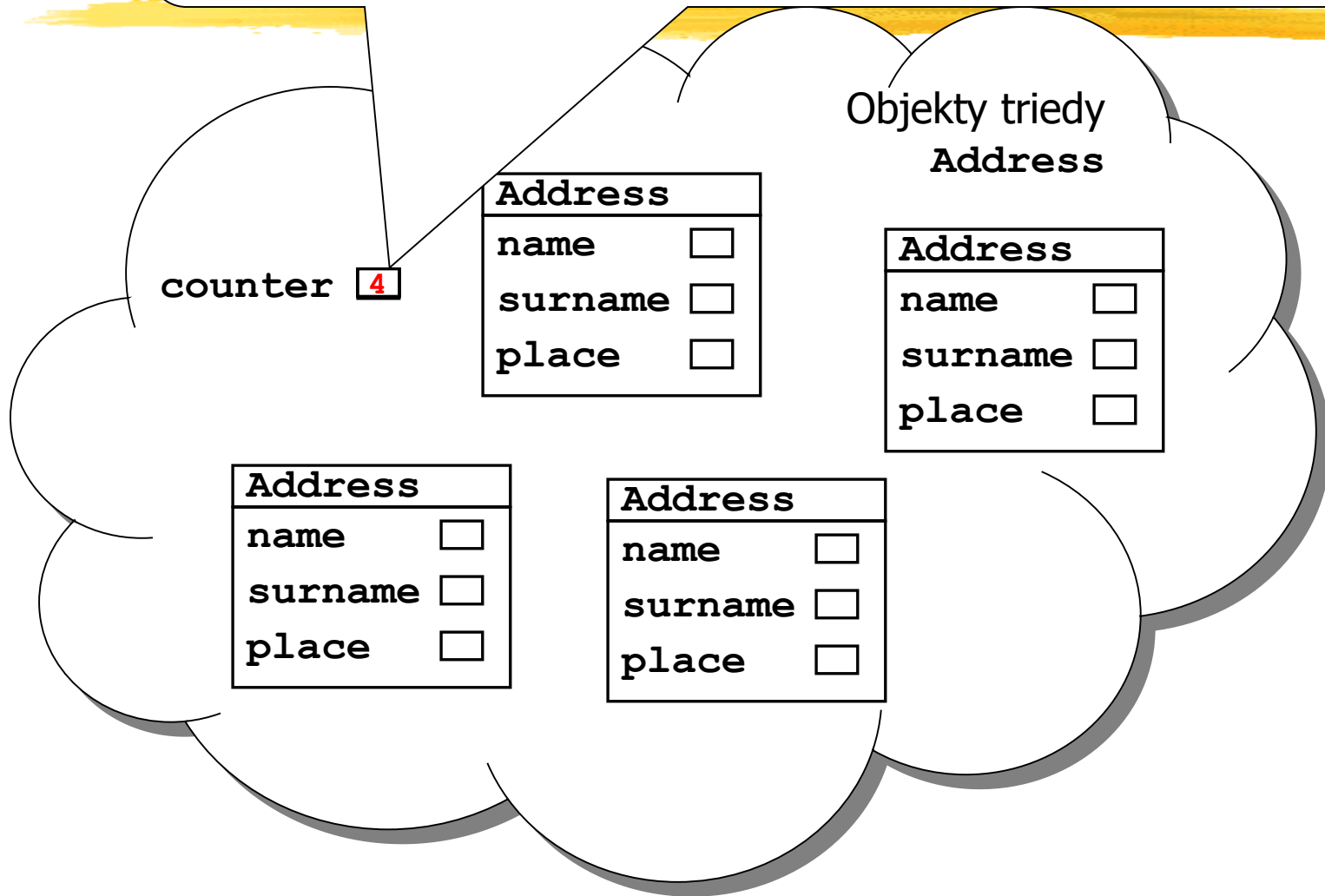
Príklad:



```
class Address
{ private String name;
  private String surname;
  private String place;
  private static int counter = 0;

  Address(String name, String surname, String place)
  { this.name = name;
    this.vorname = vorname;
    this.ort = ort;
    counter = counter + 1; // Počíta počet objektov
                          // typu Adress
  }
}
```

Statický atribút triedy existuje iba jedenkrát pre celú triedu (preto statický, nevytvára sa dynamicky pri vytváraní objektov);



Použitie statických atribútov



- V statickej premennej môžu byť uložené informácie spoločné pre všetky objekty triedy, teda informácie, ktoré jednotlivé objekty zdieľajú.

Prístup ku statickým atribútom

- Vo vnútri metódy triedy sa môže pristupovať k jej statickým atribútom použitím ich mena:
counter
- Ak máme vytvorený objekt triedy, k statickému atribútu triedy môžeme pristupovať pomocou bodkovej notácie:
adr.counter
- K statickému atribútu triedy však môžeme pristupovať aj použitím názvu triedy nasledovaným bodkou a názvom statického atribútu:
Address.counter
Math.PI

Statické metódy



- Metódy definované použitím kľúčového slova `static`.
- Tak ako statické premenné, statické metódy sú nezávislé od konkrétneho objektu triedy.
- Preto v nich môžeme pristupovať priamo iba k statickým premenným triedy.

Volanie statickej metódy

- Z metódy triedy môžeme zavolať inú metódu triedy použitím jej mena nasledovaným hodnotami parametrov.

Štandardne je statická metóda volaná použitím mena triedy nasledovaného bodkou a menom metódy:

```
Zklavesnice.readString("Zadaj meno");
```

```
Math.max(a,b);
```

(Samozrejme, ak máme definovaný objekt triedy, môžeme použiť namiesto mena triedy adresu objektu, je to však neobvyklé)

Statická trieda main



- Programy v Jave, ktoré sme doteraz používali sú tiež špeciálne triedy, ktoré majú iba statické metódy, medzi nimi špeciálnu statickú metódu:

```
public static void main(String[] args)
```

Každá trieda, ktorá túto metódu obsahuje sa dá spustiť po skompilovaní, pritom sa zavolá práve metóda `main`. Táto trieda môže mať aj nestatické atribúty a metódy.