Programovanie, algoritmy, zložitosť / ÚINF/PAZ1c

PAZ1C

Róbert Novotný
robert.novotny@upjs.sk

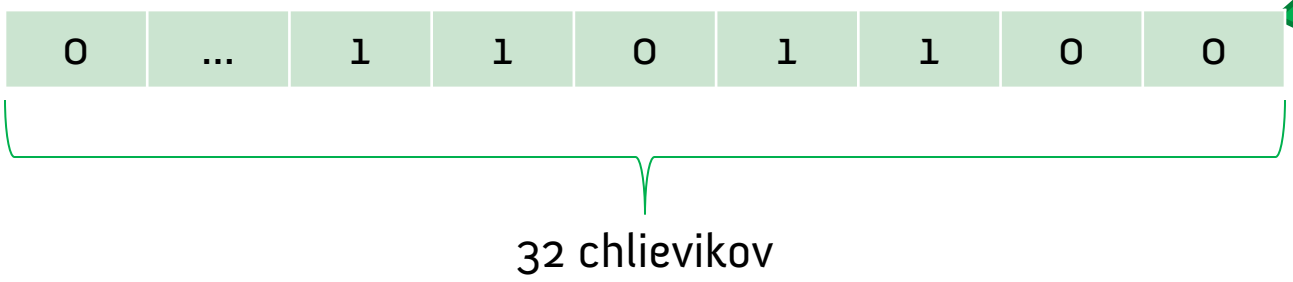
21. 10. 2009

Reprezentácia dátových typov v pamäti počítača

- alias *„Bratia == a equals() zasahujú“*
- jestvujú dva druhy dátových typov: primitívy a objekty
  - primitívy: `int`, `boolean`, `float`, `double`,...
    - dátové typy začínajúce malým písmenom
  - objekty: `String`, `Pes`,...
    - začínajúce veľkým písmenom

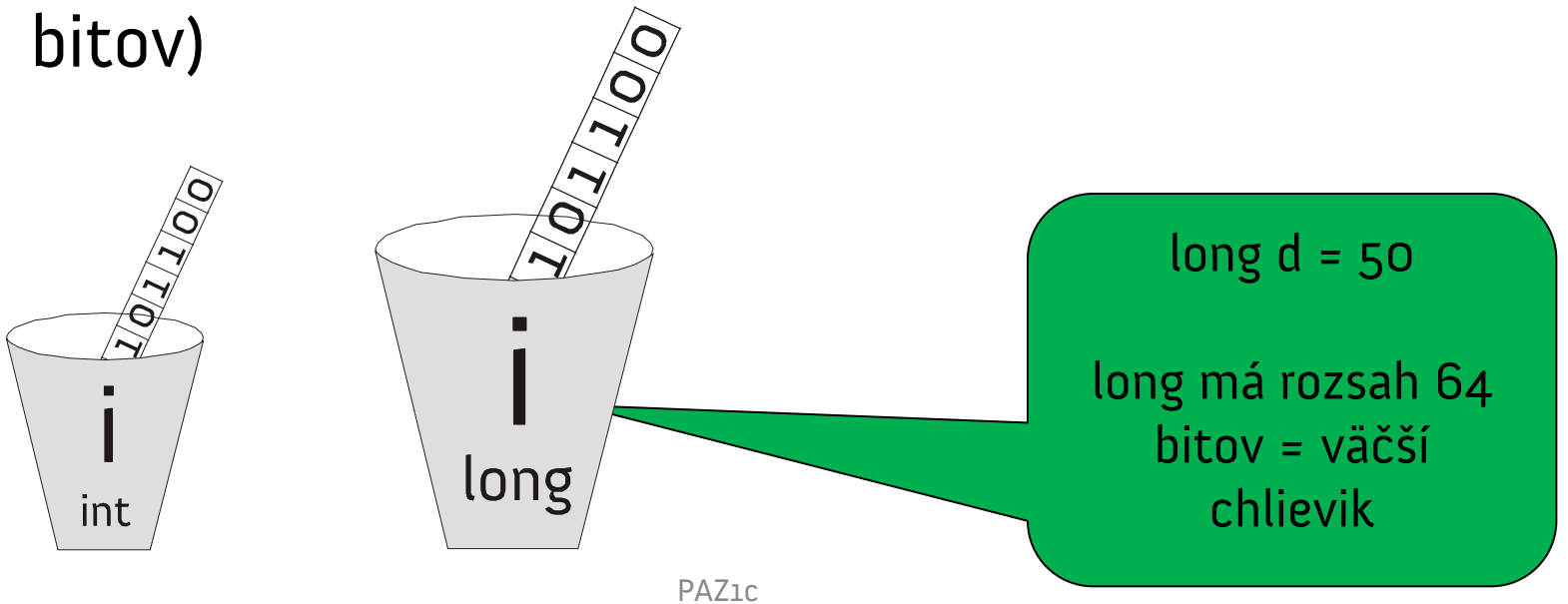
Reprezentácia primitívov v pamäti počítača

- primitívy: presne ako v Pascale
  - premenná je chlievik v pamäti, ktorý má
    - názov (`i`)
    - dátový typ (`int`)
    - veľkosť podľa dátového typu (`int`: 32 bitov)
- príklad: `int i = 52`. V binárnom kóde: `110100`



Reprezentácia primitívov v pamäti počítača

- premenná je chlievik v pamäti, ktorý má
  - názov (`i`)
  - dátový typ (`int`)
  - veľkosť (rozsah) podľa dátového typu (`int`: 32 bitov)



Reprezentácia primitívov v pamäti počítača

dátový typ	veľkosť
int	32 bitov
float	32 bitov
boolean	ťažko povedať, povedzme 1 bit
double	64 bitov
byte	8 bitov

- porovnanie primitívov: výhradne cez `==`
- porovnajú sa chlieviky bit po bite.
  - `110100 (50) == 110100 (50)`
  - `110101 (51) != 110100 (50)`

Reprezentácia objektov v pamäti

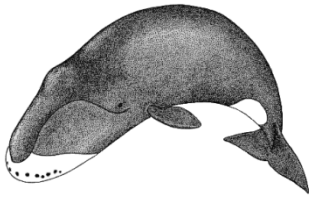
- Objekt je premenná typu špecifikovaného triedou objektu.
- premenná je chlievik v pamäti, ktorý má
  - názov (`i`)
  - dátový typ (`int`)
  - **veľkosť** podľa dátového typu
- Veľkosť?
  - aký veľký je `String`? A `Pes`? A `Veľryba`?

Je Veľryba väčšia než Mravec?

- nevieme, aký veľký je objekt
- objekty nemôžeme natlačiť do chlievika
  - čo keď sa nezmestia?
  - *jak dostat velrybu do pohárku*
  - nemôžeme mať nekonečne veľký chlievik

Riešenie

- smerní...ehm, referencie



<

Všetky objekty sú na kope... teda halde

„Urob si haldu v pamäti počítača, v halde urob priehrady a zvnútra i zvonka ich vymaž smolou! A postav ju takto: tristo MB bude jej dĺžka, päťdesiat MB jej šírka a tridsať MB jej výška.

Do korába vojdeš ty i tvoji synovia, tvoja žena aj ženy tvojich synov s tebou.

Zo všetkých vtákov podľa svojho druhu, z dobytka podľa svojho druhu a z plazov podľa svojho druhu vojdú po dvoch do korába s tebou, aby mohli žiť."

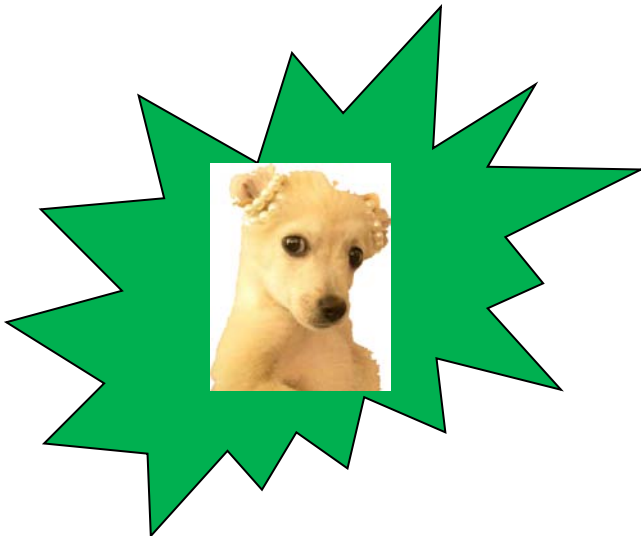
– IT Genesis

Všetky objekty sú na kope... teda halde

- **halda** (*heap*) je priestor v pamäti určený pre objekty
  - nemýliť si s heapsortom (triedenie haldovaním)!
- je spravovaný automaticky Javou
- prostý programátor nevie o existencii haldy
- na halde sa dejú kadejaké zverstvá
  - automatické uvoľnovanie pamäte (*Garbage Collection*)
  - o tom však neskôr

Detaily v útrobách psa

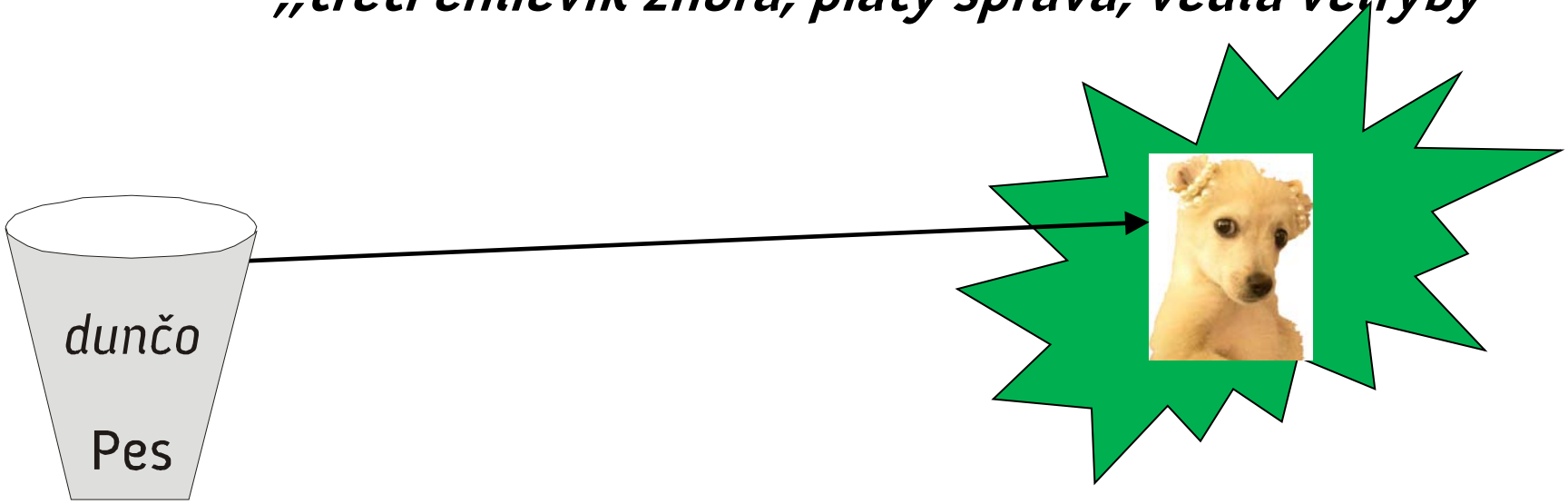
- `Pes dunčo = new Pes()`
- `Pes dunčo;`
  - vytvorí sa nové premenná dunčo typu `Pes`
- `new Pes()`
  - na halde sa vytvorí dostatok pamäte pre novú inštanciu



Všetky objekty sú na kope... teda halde

- `Pes dunčo = new Pes()`
- premenná dunčo je nasmerovaná na inštanciu psa na halde. Bude obsahovať adresu inštancie na halde

„tretí chlievik zhora, piaty sprava, vedľa veľryby"



Adresa ja, adresa ty...

- premenná dunčo obsahuje adresu inštancie na halde
- to je presne idea smerníkov
- našťastie:
  - smerníky sú v pozadí
  - užívateľ ich nevidí
    - ani nechce vidieť
  - žiadne `^.` ako v Pascale, ani `*` ako v C

Podobenstvo s diaľkovým ovládaním

- premenná Pes obsahuje „diaľkové ovládanie" inštancie na halde

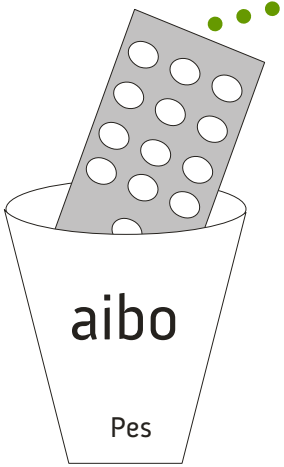
```
Pes aibo = new Pes();
```

1. **`Pes aibo`** ... vytvoríme novú premennú

   teda pohárik s diaľkovým ovládaním

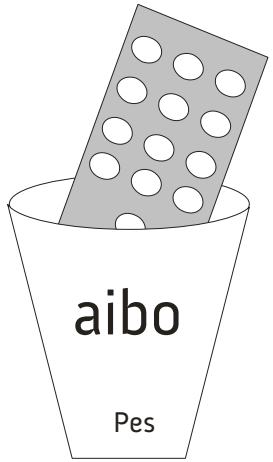
2. **`new Pes()`** ... vytvorenie novej inštancie na halde

3. **`priradenie`** .. diaľkové ovládanie naprogramujeme na ovládanie konkrétnej inštancie (teda Aiba).



Podobenstvo s diaľkovým ovládaním

- Ak pohárik `int` má veľkosť 32 bitov, akú veľkosť má pohárik typu `Pes`?
- Nevedno, ale ani nás to netrápi.
  - JDK od Sunu: 64 bitov
  - Java od Janka Hraška:
    - 64 bitov na obed
    - 32 bitov v noci

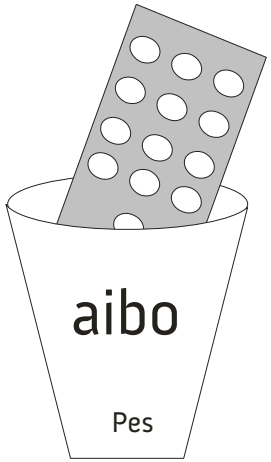


Otázka k diaľkovým ovládaniam

- Ak nadeklarujem premennú a nepriradím jej nič, koho riadi diaľkové ovládanie?

```
Pes aibo;
```

- Premenná, ktorej nebola priradená žiadna inštancia ukazuje na `null`.
  - `null` - smerník, ktorý ukazuje nikam.
- Ak premenná ukazuje na ***null***, mám diaľkové ovládanie, ale nemám k nemu televízor.



Dôsledky diaľkového ovládania

- Čo spraví nasledovný kód?

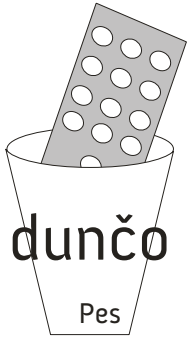
```
Pes dunčo = new Pes();
dunčo.setRasa("čuvač„);
dunčo.setVek(25);
System.out.println(dunčo.getVek());

Pes aibo = dunčo;
aibo.setVek(35);
System.out.println(aibo.getVek());

System.out.println(dunčo.getVek());
```

```
25
35
```

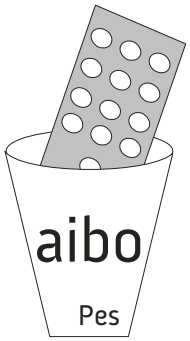
```
35
```

Ale prrrrečo?


```
Pes dunčo = new Pes();
```



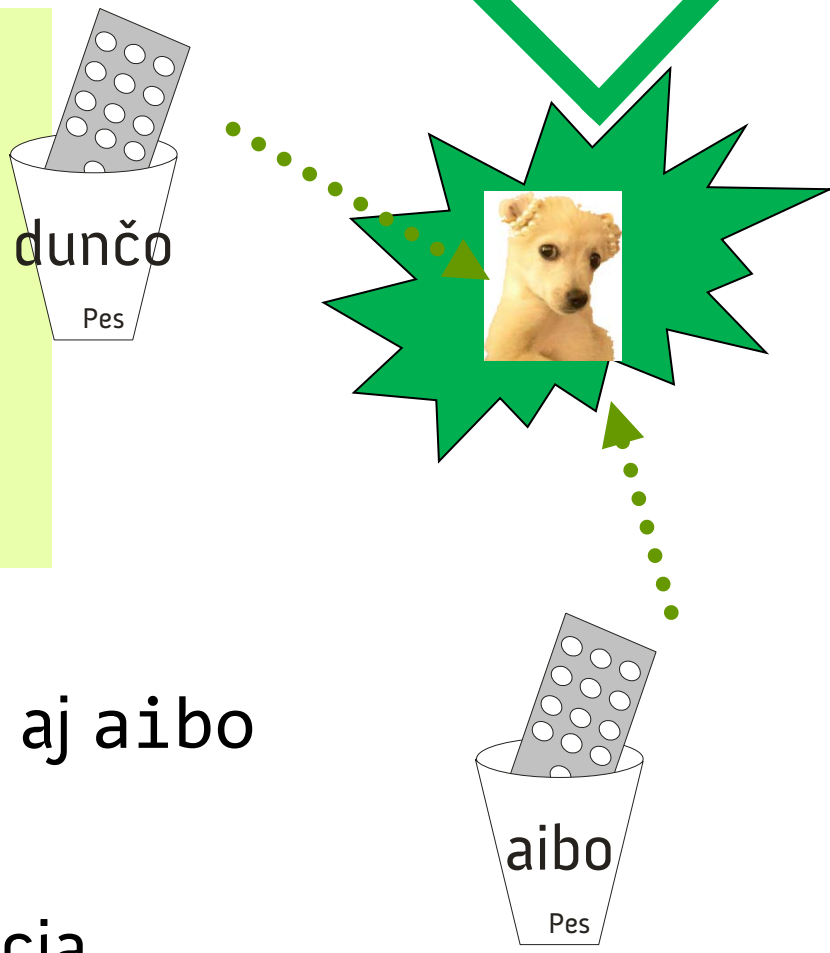
```
Pes aibo  = dunčo
```



- obe diaľkové ovládania riadia toho istého psa
- ak zmeníme vek pomocou **aiba**, zmení sa aj vek pre **dunča**

Dôsledky diaľkového ovládania

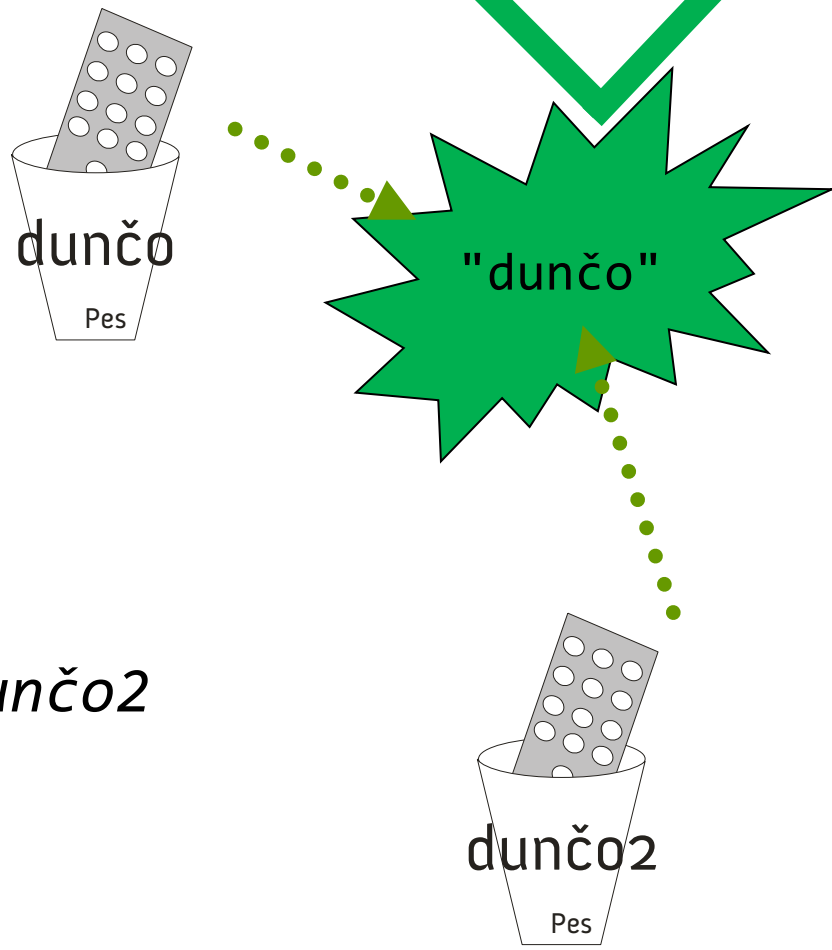

```
Pes dunčo = new Pes();
Pes aibo = dunčo;

if(dunčo == aibo) {
    //platí
}
```



- Podmienka platí, lebo dunčo aj aibo ukazujú na toho istého psa.
- Toto je však výnimočná situácia.

Dôsledky diaľkového ovládania

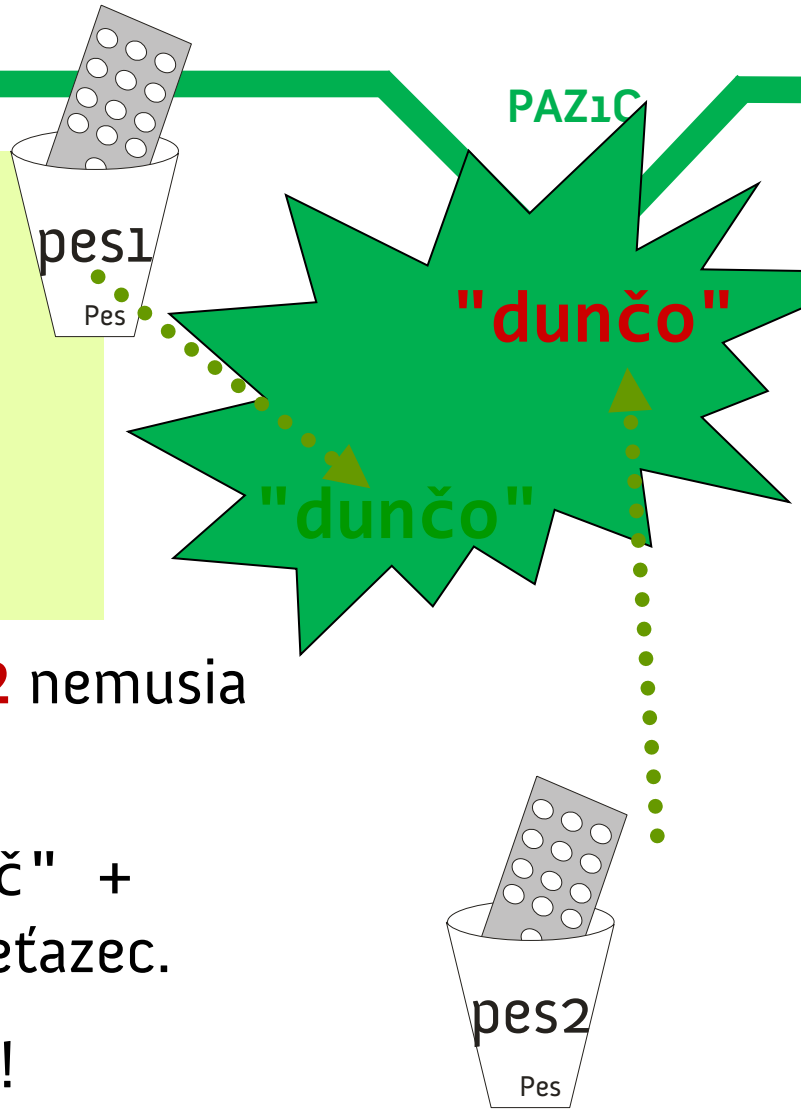

```
String dunčo = "dunčo";
String dunčo2 = dunčo;

if(dunčo == dunčo2) {
    //platí
}
```



- Podmienka platí, lebo *dunčo* aj *dunčo2* ukazujú na ten istý reťazec.

Dôsledky diaľkového ovládania

```
String pes1 = "dunčo";
String pes2 = "dunč" + "o";

if(pes1 == pes2) {
      //ráno platí, večer už nie
}
```



- Podmienka platí, lebo **pes1** aj **pes2** nemusia ukazovať na ten istý reťazec.
- Kompilátor nemusí vedieť, že "dunč" + "o" má nasmerovať na existujúci reťazec.
- Preto porovnávame cez **equals()** !

Zásada s veľkým Z

Objekty porovnávame cez equals()!

- názov typu objektu sa začína veľkým písmenom
  - Stringy sú objekty!

Primitívne typy porovnávame cez ==!

- názov primitívu sa začína malým písmenom
  - primitívy nie je možné porovnávať cez equals()!
  - primitív nemá metódy, nastane kompilačná chyba

Zásada s veľkým Z2

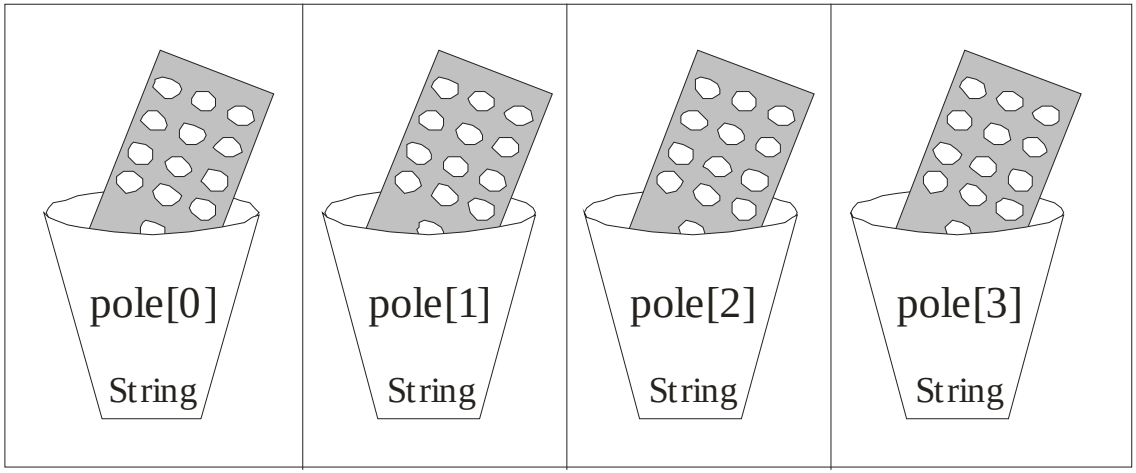
Objekty porovnávame cez == jedine v prípade, že ich porovnávame s `null`.

```
Pes pes; // v psovi je null
if(pes == null) {
  System.out.println("Kde je pes?“);
}
```

- nepíšeme ~~`pes.equals(null)`~~
  - vedie to k chybe
  - **null** znamená *nič* a *nič* nemá žiadne schopnosti (= žiadne metódy!)

Dôsledky diaľkového ovládania - polia


```
String[] pole = new String[4];
```



- Máme pole štyroch pohárikov s diaľkovými ovládaniami, ktoré neriadia žiaden objekt (neukazujú nikam)
- Každý z prvkov má hodnotu **null**.
- Dôsledok: **`pole[0].length()`** = **NullPointerException** = **výbuch**
  - snažíme sa volať metódu na neexistujúcom objekte!

Dôsledky diaľkového ovládania – polia a objekty


```
Pes dunčo = new Pes();
Pes[] psi = new Pes[2];
psi[0] = dunčo;
psi[0].setVek(4);
System.out.println(psi[0].getVek());
System.out.println(dunčo.getVek());
```