Překrývání metod

Table of Contents

Třída `Object`

- I když v Javě vytvoříme prázdnou třídu, obsahuje 'skryté' metody.
- Je to technicky tím, že všechny třídy dědí přímo či nepřímo z třídy Object a odtud ty metody jsou.

```
public class Person { }

Person p = new Person();
p.toString(); // ???
```

- Seznam všech vestavěných metod najdete v javadocu třídy `Object`.

Některé metody třídy `Object`

- `getClass()` — vrátí název třídy
- `equals(⋯)` — porovná objekt s jiným, `hashCode()` — vrátí haš
- `clone()` — vytvoří identickou kopii objektu (*deep copy*)
  - tahle metoda však může způsobovat problémy (vysvětlíme později)
  - proto ji **nepoužívejte**
- `toString()` — vrátí textovou reprezentaci objektu

```
Person p = new Person();
System.out.println(p);
// it simply does this - for non-null p:
System.out.println(p.toString());
```

Překrytí metody

- Podívejme se blíž na metodu `String toString()`.
- Co kdybychom chtěli její chování změnit?
- Zkusme v naší třídě implementovat metodu se stejným jménem:

```
public class Person {
  public String toString() {
    return "it works";
  }
}
Person p = new Person();
System.out.println(p); // it works
```

Metoda `toString()`

- Javadoc říká, že každá třída by měla tuhle metodu překrýt.
- Co se stane, když ji nepřekryjeme a přesto ji zavoláme?
- Použije se výchozí implementace z třídy `Object`:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

```
Person p = new Person();
System.out.println(p); // Person@14ae5a5
```

- Vypíše se jméno třídy, zavináč, a pak hexadecimálně nějaký podivný hash.

Anotace `@Override` — motivace

- Bylo by fajn mít kontrolu nad tím, že překrýváme skutečně existující metodu.
- Najdete chybu?

```
public class Person {
  public String toStrng() {
    return "not working";
  }
}
Person p = new Person();
System.out.println(p); // Person@14ae5a5
```

Anotace `@Override`

- Použijeme proto **anotaci**, která kompilátoru říká: *přepisuji existující metodu*.
- Anotace se píše před definici metody:

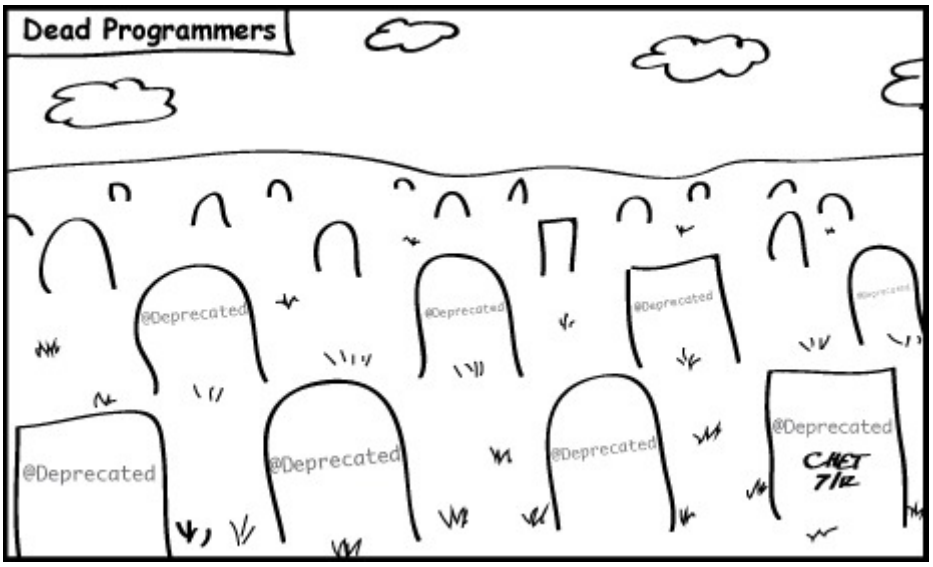
```
@Override
public String toString() {
  return "it works again";
}
```

- Kdybychom udělali chybu např. v názvu překrývané metody, kód by nešel přeložit.

Vždy používejte anotaci `@Override`, když přepisujete metodu.

Jiné notace

Existují i jiné anotace, například `@Deprecated` naznačuje, že by se daná věc už neměla používat.



`Objects.toString`

- Například zřetězení `"Osoba " + p` se kód přeloží skoro jako `"Osoba " + p.toString()`
- s výjimkou `p == null`, kdy vrátí `"Osoba null"`.
- Přímé volání `o.toString()` tedy vyžaduje test na nulovost `o` anebo
- bezpečně a bez testů použít `Objects.toString(o)`,
- nebo `Objects.toString(o, "žádná")`, kdy vrátí v případě `o == null` náhradní text `"žádná"`.