czechitas

Java 1 • lekce 11

Filip Jirsák

26. 3. 2024

online

Statické fieldy a metody

Fieldy a metody spojené s celou třídou, ne s konkrétní instancí. Přistupuje se k nim přes název třídy.

- klíčové slovo `static`

```
public class Person {
  private static int count;

  public Person() {
    Person.count++;
  }
}
```

Třídy a rozhraní

- třída (`class`) – definuje data společně s metodami, které s daty pracují
- rozhraní (`interface`) – popis toho, jaké metody má mít třída
- `default` metody na rozhraní – obsahují implementaci, která ale může volat jen jiné metody daného rozhraní, nemůže používat data
- `enum` – speciální typ třídy reprezentující výšt hodnot (např. roční období, dny v týdnu)
- `record` – speciální typ třídy, slouží priárně k uchování dat, která nelze měnit
- anotace (`@interface`) – speciální typ rozhraní, slouží pro značkování tříd, metod a konsturktoů, fieldů, parametrů a návratových typů metod

Dědičnost

- Třída může dědit (`extends`) od jiné třídy – pouze od jedné
- Třída může implementovat (`implements`) rozhraní – může implementovat více rozhraní

```
public class AbstractList implements List {

}

public class ArrayList extends AbstractList {

}
```

Abstraktní metody a třídy

- Abstraktní metoda není implementovaná, pouze definuje rozhraní.
- Některé její metody jsou abstraktní (neimplementované), proto nelze vytvářet instance abstraktní třídy.
- Abstraktní třída je podobná rozhraní, ale může obsahovat data (fieldy) a mít implementované některé metody.

```java
public abstract class Tvar {
  private int plocha;

  public abstract void nakresli();
}

public class Ctverec extends Tvar {

  @Override
  public void nakresli() {
    // nakreslí čtverec
  }
}

public class Kruh extends Tvar {

  @Override
  public void nakresli() {
    // nakreslí kruh
  }
}
```

Modifikátory přístupu

- `public` – přístupné ze všech tříd
- `protected` – přístup ze tříd ze stejného balíku a ze zděděných tříd
- *bez uvedení* (package protected) – přístup ze tříd ze stejného balíku
- `private` – přístup pouze z třídy, kde je field/metoda deklarována

Modifikátor `final`

Označuje neměnný field, metodu nebo proměnnou.

- **field** – konstanta, nelze měnit jeho hodnotu (je zvykem psát jména jako CAMEL_CASE)
- **proměnná** – nelze měnit její hodnotu
- **metoda** – nelze ji přetížit v potomcích

```
class Test {
  private final static String ADRESA_WEBU = "https://www.czechitas.cz";

  public int prumer(int a, int b) {
    final int soucet = a + b;
    return soucet / 2;
  }

  public final void vypsatAdresuWebu() {
    System.out.println(ADRESA_WEBU);
  }

}
```

Vnitřní třídy

Vnitřní třída – třída, která je definovaná uvnitř jiné třídy, protože je s ní nějak svázána. Má přístup k fieldům a metodám dané třídy.

Pokud je potřeba přístup k vnitřní třídě z venku, použije se název vnější třídy, za něj se přidá tečka a za ni název vnitřní třídy. Např. `` `cz.czechitas.Mapa.PolozkaMapy` ``.

```
package cz.czechitas;

public class Mapa {

  private List<PolozkaMapy> polozky = new ArrayList<PolozkaMapy>();

  public class PolozkaMapy {
    private String klic;
    private String hodnota;
  }
}
```

Překrývání metod a konstruktorů (override)

Překrytá metoda nebo konstruktor je metoda, která má stejnou signaturu (název, počet parametrů a jejich typ), jako metoda v předkovi, ale ke implementovaná v potomkovi jinak. Pro přístup k metodě/konstruktoru předka se používá klíčové slovo `super`.

Překryté metody se označují anotací `@Override`.

Např. pokud implementujeme v naší třídě metodu `toString()`, překrýváme metodu `Object.toString()`.

Přetěžování metod a konstruktorů (overload)

Přetížení metody znamená, že máme v jedné třídě více metod, které mají stejný název, ale liší se počtem nebo typem parametrů.

Např. metody v rozhraní `List` jsou metody `add(Object)` a `add(int, Object)` přetížené.

```java
public class Osoba {
  public Osoba(String jmeno, String prijmeni) {
    this.jmeno = jmeno;
    this.prijmeni = prijmeni;
  }

  @Override  // Metoda je zděděná z java.lang.Object
  public String toString() {
    return jmeno + " " + prijmeni;
  }

}

public class Zamestnanec extends Osoba {
  public Zamestnanec(String jmeno, String prijmeni, String pozice) {
    super(jmeno, prijmeni);
    this.pozice = pozice;
  }

  @Override
  public String toString() {
    return super.toString() + " (" + pozice + ")";
  }
}
```

Sealed classes

- Detailnější specifikace dědičnosti
- Může se použít ve vlkých projektech nebo knihovnách