Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 18

1. Records
2. Chyby pri kompilácii
3. Chyby pri behu aplikácie

Records

Record je špeciálny typ triedy určený na nesenie dát (data carrier)

```
public record Osoba(String meno, String priezvisko,
Pohlavie pohlavie, LocalDate datumNarodenia) {}
```

Records odstraňuje prebytočný kód (anglicky boilerplate), prináša jednoduchosť a zvyšujú spoľahlivosť kódu. `record` automaticky generuje:

- private final atribúty
- parametrický konštruktor
- gettre (bez get prefixu)
- metódy `equals()`, `hashCode()` a `toString()`

Trieda vygenerovaná z Recordu

```
public class Osoba extends java.lang.Record {
    private final String meno;
    private final String priezvisko;
    private final Pohlavie pohlavie;
    private final LocalDate datumNarodenia;
    public SimpleOsoba(String meno, String priezvisko, Pohlavie pohlavie, LocalDate datumNarodenia) {
        this.meno = meno;
        this.priezvisko = priezvisko;
        this.pohlavie = pohlavie;
        this.datumNarodenia = datumNarodenia;
    }
    public String meno() {return meno;}
    public String priezvisko() {return priezvisko;}
    public Pohlavie pohlavie() {return pohlavie;}
    public LocalDate datumNarodenia() {return datumNarodenia;}
    @Override
    public boolean equals(Osoba other) {
        return this.meno.equals(other.meno) && this.priezvisko.equals(other.priezvisko)
            && this.pohlavie.equals(other.pohlavie) && this.datumNarodenia.equals(other.datumNarodenia);
    }
    public int hashCode() {
        return java.util.Objects.hash(meno, priezvisko, pohlavie, datumNarodenia);
    }
    public String toString() {
        return "...výpis hodnôt";
    }
}
```

Records

Recordy sú nemenné:

- všetky polia sú `final`
- neexistujú settre
- stav objektu sa po vytvorení nemení

Recordy sa nededia:

- record je definovaný ako `final`
- record dedí iba z triedy `java.lang.Record`
- record ale môže implementovať akékoľvek rozhrania

Records

Do recordov môžete pridávať vlastné metódy a statické atribúty.

Podporujú tiež vlastný konštruktor v tzv. kompaktom formáte, kedy nemusíme písať argumenty konštruktora:

```
public record Interval(int od, int do) {
  // kompaktný konštruktor
  public Interval {
    if (od > do)
      throw new IllegalArgumentException("od je > ako do");
  }

  public int dlzka() {
    return do - od + 1;
  }
}
```

Records

Recordy dobre spolupracujú s knižnicou Jackson, ktorá umožňuje načítať hodnoty z JSON alebo CSV súborov priamo do recordov.

```
@JsonPropertyOrder({"meno", "priezvisko", "pohlavie"})
public record RecordOsoba
    (@JsonProperty("meno") String meno,
     @JsonProperty("priezvisko") String priezvisko,
     @JsonProperty("pohlavie") Pohlavie pohlavie) {

  public String toString() {
    return meno + " " + priezvisko;
  }
}
```

Chyby pri kompilácii

Veľké množstvo chýb nám dokáže odhaliť kompilátor a teda prídeme na nich ešte pred tým, ako sa program spustí.

Najčastejším zdrojom týchto chýb je kopírovanie kódu z webových stránok alebo z asistentov umelej inteligencie.

Automatické formátovanie kódu (*Code-Reformat Code*) pomocou `Ctrl-Alt-L` nám pomôže nájsť chybu.

Typické kompilačné chyby:

- *cannot find symbol* - Chýba import triedy alebo sme v názve urobili preklep
- *class, interface, enum, or record expected* - Máme niekde zátvorku navyše
- *reached end of file while parsing* - Niekde nám zátvorka chýba
- *illegal start of expression* - Chýbajúca zátvorka

Chyby pri behu aplikácie

Pri vyhodení výnimky program vypíše **stacktrace**

Stacktrace je výpis volaní metód, pomocou ktorého vieme nájsť zdroj chyby, teda metódu, ktorá vyhodila výnimku

Medzi najčastejšie výnimky počas behu programu patria `NullPointerException` a `ClassCastException`.

V dokumentácii sú uvedené výnimky, ktoré metóda môže vyhodiť.

Hľadanie príčine v stacktrace

- riadky, ktoré sú z našich tried (Osoba.java)
- prvé riadky začínajúce s `at`
- skutočná príčina je často v sekcii `Caused by`, ktorá obsahuje výpis vnorenej výnimky

Stacktrace

```
Exception in thread "main" com.fasterxml.jackson.databind.JsonMappingException: Cannot invoke
"String.toUpperCase()" because "this.priezvisko" is null (through reference chain:
java.util.ArrayList[0]->sk.spse.jackson.Osoba["priezvisko"])
    at com.jackson.databind.JsonMappingException.wrapWithPath(JsonMappingException.java:400)
    at com.jackson.databind.JsonMappingException.wrapWithPath(JsonMappingException.java:359)
    at com.jackson.databind.ser.std.StdSerializer.wrapAndThrow(StdSerializer.java:324)
    at com.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:765)
    at com.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:183)
    at com.jackson.databind.ser.IndexedListSerializer.serializeContents(IndexedListSerializer.java:119)
    at com.jackson.databind.ser.impl.IndexedListSerializer.serialize(IndexedListSerializer.java:79)
    at com.jackson.databind.ser.impl.IndexedListSerializer.serialize(IndexedListSerializer.java:18)
    at com.jackson.databind.ser.DefaultSerializerProvider._serialize(DefaultSerializerProvider.java:503)
    at com.jackson.databind.ser.serializeValue(DefaultSerializerProvider.java:342)
    at com.jackson.databind.ObjectMapper._writeValueAndClose(ObjectMapper.java:4926)
    at com.jackson.databind.ObjectMapper.writeValue(ObjectMapper.java:4088)
    at sk.spse.jackson.Json.main(Json.java:42)
Caused by: java.lang.NullPointerException: Cannot invoke
"String.toUpperCase()" because "this.priezvisko" is null
    at sk.spse.jackson.Osoba.getPriezvisko(Osoba.java:50)
    at java.base/jdk.internal.reflect.invoke(DirectMethodHandleAccessor.java:103)
    at java.base/java.lang.reflect.Method.invoke(Method.java:580)
    at com.fasterxml.jackson.databind.ser.serializeAsField(BeanPropertyWriter.java:688)
    at com.fasterxml.jackson.databind.ser.std.serializeFields(BeanSerializerBase.java:760)
    ... 9 more
```

Chyby pri behu aplikácie

Niekedy nevyhodí výnimku, ale aj tak nefunguje správne, nereaguje podľa očakávaní, alebo nevyprodukuje požadovaný výstup

Vtedy sa odporúča použiť debugovacie nástroje IDE, alebo pridať do kódu stručný výpis pomocou `System.out.println` na tie miesta, kde tušíme chybu.

Debugovanie v IDEA

Debug režim – spustenie programu pod kontrolou debuggeru (Shift + F9)

Breakpointy – zastavenie programu na riadku kódu

Ovládanie behu programu

- F8 – Step Over (vykoná riadok)
- F7 – Step Into (vojde do metódy)
- F9 – Resume (pokračuje)

Variables panel – aktuálne hodnoty premenných

Watches / Evaluate Expression (Alt + F8) – sledovanie výrazov a metód

Call Stack – poradie volaných metód

Break on Exception – zastaví program v mieste vzniku chyby