Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 10

1. Zapuzdrenie
2. Triedny Invariant
3. Nemennosť
4. Defenzívne kopírovanie

Zapuzdrenie

Zabalenie dát (atribútov) a metód do jednej komponenty – objektu

- Zvyšuje bezpečnosť
- Zlepšuje modularitu
- Jednoduchšie sa udržiava kód

Hlavné ciele zapuzdrenia sú **ochrana dát**, **abstrakcia** a **flexibilita**

Zapuzdrenie

Techniky zapuzdrenia

- Modifikátory prístupu
- Getter a setter metódy
- Skrytie implementačných detailov do súkromných metód

Privátne konštruktory a továrenske metódy

```
public class Osoba {
    private String meno; // súkromný atribút

    // Getter
    public String getMeno() {
        return meno;
    }

    // Setter
    public void setMeno(String meno) {
        // validácia vstupu
        if(meno != null && !meno.isEmpty())
            this.meno = meno;
    }
}
```

Použitie:

```
public class Main {
    public static void main(String[] args) {
        Osoba osoba = new Osoba();
        osoba.setMeno("Jano");
        System.out.println("Meno: " + osoba.getMeno());
    }
}
```

Invariant

Podmienka alebo vlastnosť, ktorá je pravdivá pre objekt počas jeho celého života

- Platí po každom skončení konštruktora
- Platí pred aj po každom verejnom volaní metódy
- Môže byť dočasne porušený vo vnútri metódy, ale musí byť obnovený pred jej ukončením

```
public class BankovyUcet {
    private double zostatok;
    private String cisloUctu;

    public BankovyUcet(String cisloUctu, double pociatocnyZostatok) {
        this.cisloUctu = cisloUctu;
        if (pociatocnyZostatok >= 0) {
            this.zostatok = pociatocnyZostatok;
        } else {
            throw new IllegalArgumentException("Záporný zostatok!");
        }
    }

    public void vloz(double suma) {
        if (suma > 0) {
            zostatok += suma;
        } else {
            throw new IllegalArgumentException("Suma má byť kladná");
        }
    }
}
```

Nemenný objekt

Nemenný - **immutable** - objekt, ktorého stav sa po vytvorení už nikdy nezmení

V nemennej triede sa používajú konštanty alebo sa neposkytnú setter metódy

```
public final class Auto {
    private final String znacka;
    private final int rok;

    public Auto(String znacka, int rok) {
        this.znacka = znacka;
        this.rok = rok;
    }

    public String getZnacka() {
        return znacka;
    }

    public int getRok() {
        return rok;
    }
}
```

Defenzívne kopírovanie

Getter metóda môže spôsobiť únik vnútorného objektu

Riešením je vrátiť vždy kópiu vnútornej hodnoty objektu

```
public class Trieda {
    private String[] ziaci = new String[30];

    // Konštruktor
    public Trieda(String... ziaci) {
        // Invariant triedy
        for (var ziak: ziaci)
            Objects.requireNonNull(ziak, "Ziak je null!");
        this.ziaci = ziaci;
    }

    // Zlý getter - vracia priamu referenciu
    public String[] getZiaci() {
        return ziaci;
    }
}
```

Zmena vnútorného atribútu:

```
Trieda trieda = new Trieda("Fero", "Jano", "Beatka");
String ziaci = trieda.getZiaci();
ziaci[0] = null; // invariant porušený
```

Riešenie je defenzívne kopírovanie:

```
public String[] getZiaci() {
    return ziaci.clone();
}
```