**Fakulta informatiky a informačných technológií**

Slovenskej technickej univerzity v Bratislave

Základy objektovo-orientovaného programovania

**Organizačný softvér tenisových turnajov**

**Použité princípy objektovo-orientovaného programovania**

Emma Macháčová

Zimný semester 2020/2021

Použité princípy obiektovo-orientovaného programovnia

[1. Dedenie 3](#_Toc56331255)

[1.1 Viacnásobné dedenie 3](#_Toc56331256)

[2. Modifikátory prístupu 3](#_Toc56331257)

[2.1 Getters & Setters 3](#_Toc56331258)

[3. Overriding 4](#_Toc56331259)

[4. Overloading 4](#_Toc56331260)

[4.1 Preťažený konštruktor 4](#_Toc56331261)

[4.2 Preťažená metóda 4](#_Toc56331262)

[5. Asociácia 4](#_Toc56331263)

[6. Agregácia 5](#_Toc56331264)

[7. Vhodná organizácia tried do balíkov 5](#_Toc56331265)

1. Dedenie

public class TurnajA **extends Turnaj** {} // z hlavného classu Turnaj

public class TurnajB extends Turnaj {}

public class TurnajC extends Turnaj {}

public class TurnajD extends Turnaj {}

1.1 Viacnásobné dedenie

public class MajstrovstvaSR **extends TurnajA** {}

1. Modifikátory prístupu

**private** static int velkostPavuka; // class Pavuk

**private** static int pocetKol;

**private** ArrayList<Kolo> kola = new ArrayList<Kolo>(0);

**protected** Kategoria kategoria; // class Turnaj

**protected** int pocetKurtov;

**protected** int pocetDni;

2.1 Getters & Setters

//metóda na získanie private premennej

public static int **getVelkostPavuka**() {

return velkostPavuka;

}

//metóda na nastavenie private premennej

public static void **setVelkostPavuka**(int velkostPavuka) {

Pavuk.velkostPavuka = velkostPavuka;

}

1. Overriding

Je to metóda classu MajstrovstvaSR, ktorá ju zdedí od TurnajaA ktorý ju zdedí z classu Turnaj, a prekonáva metódu z classu Turnaj tým, že nepotrebuje vstup dátumu

public static **Turnaj vytvorTurnaj(char typ)** {

Date datum = new Date();

datum.setMonth(3);

datum.setDate(12);

datum.setHours(8);

datum.setMinutes(0);

datum.setSeconds(0);

ArrayList<Hrac> hrac = Hrac.nacitajHracov(....);

Pavuk pavuk;

MajstrovstvaSR MS = new MajstrovstvaSR(....);

pavuk = new Pavuk(velkostPavuka,MS);

MS.setPavuk(pavuk);

return MS;

}

1. Overloading

4.1 Preťažený konštruktor

// class Kategoria

public **Kategoria()** {}

public **Kategoria (char kategoria, boolean leto)** {}

4.2 Preťažená metóda

// class Turnaj

public static Turnaj **vytvorTurnaj()** {}

public static Turnaj **vytvorTurnaj(Turnaj truenaj)** throws CloneNotSupportedException {

Turnaj turnaj = truenaj.clone();

return turnaj;

}

1. Asociácia

**Turnaj** má kategóriu(**Kategoria**) ale viacero turnajov môže mať tú istú Kategóriu pričom zmena v turnaji ovplyvní kategóriu ale zmena v kategórii neovplyvní turnaj

1. Agregácia

**Turnaj** má hráčov (**Hrac**) a s hráčmi môže byt manipulované len cez objekt Turnaj

1. Vhodná organizácia tried do balíkov

* **com.organizer**
  + **core**
    - (class DataHandler)
    - (Main)
  + **triedy**
    - **pavuk**
      * (class Kolo)
      * (class Pavuk)
    - **turnaje**
      * (class TurnajA)
      * (class TurnajB)
      * (class TurnajC)
      * (class TurnajD)
      * (class MajstrovstvaSR)
    - (class Den)
    - (class Hrac)
    - (class Kategoria)
    - (class Kurt)
    - (class Turnaj)
    - (class Zapas)