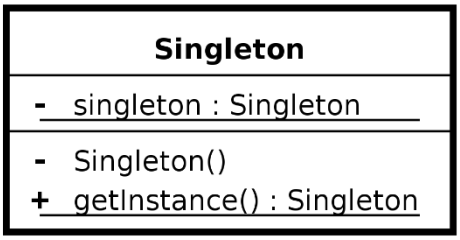
Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

1. (7 b) Vysvetlite použitie návrhového vzoru Singleton, ktorý ste implementovali vo vlastnom riešení, prípadne ak ste ho neimplementovali vysvetlite potenciálny účel jeho použitia vzhľadom na váš projekt ako aj jeho implementáciu. Uveďte zároveň, kedy skutočne vytvárate jedinú inštanciu vašej triedy vo vašom riešení.

Návrhový vzor singleton sa používa keď triedu na to, aby vytvorila iba jedinú inštanciu v jednom čase – tento návrhový vzor by som použila v rámci inštancie Turnaj, pretože sa vytvára iba jeden pri celom behu programu

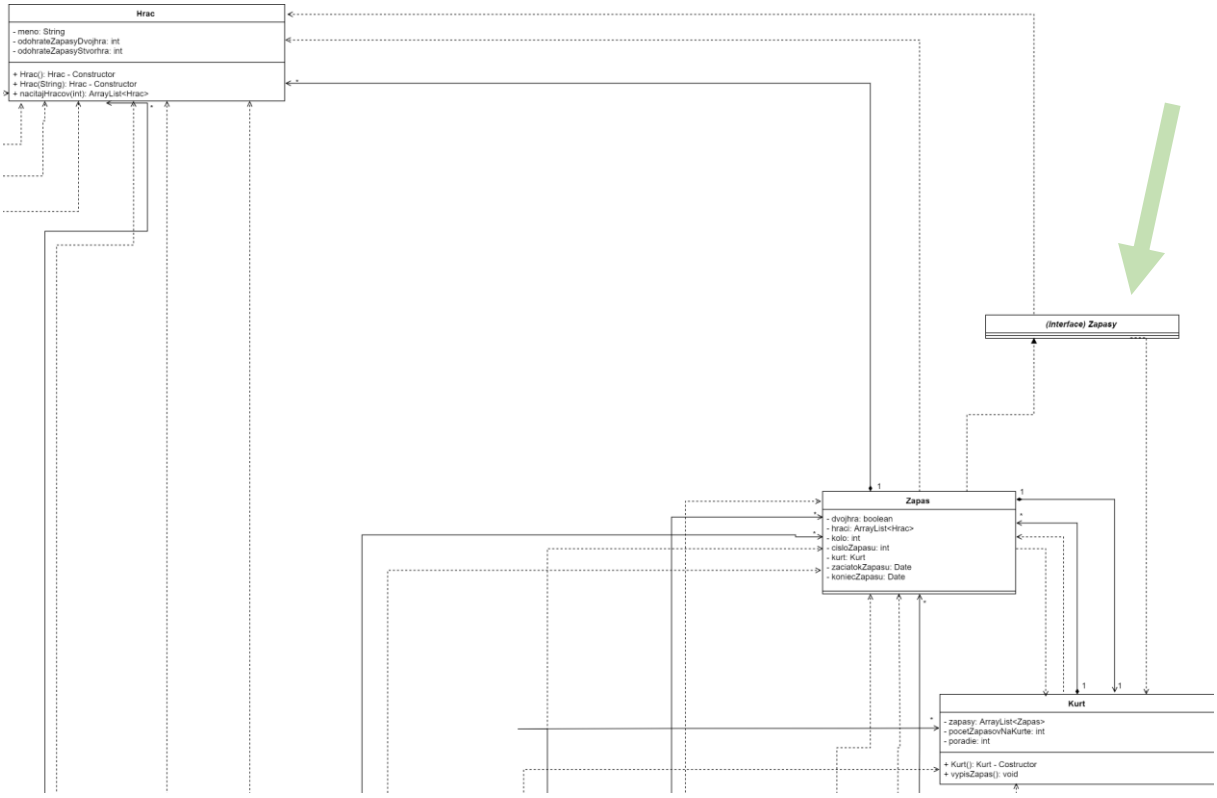


implementacia napr.

```
private static Turnaj instancia;
public static Turnaj getInstance() {
  if (instancia == null) {
    instancia = new Turnaj();
    }
}
return instancia;
```

Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

2. (7 b) Diagramom tried v UML vyjadrite vzťah vybraného rozhrania z vášho projektu, triedy, ktorá ho implementuje, a triedy ktorá ho používa. V triedach a v rozhraní stačí uviesť len nevyhnutné prvky z hľadiska otázky. Poskytnite relevantný kód z vašej implementácie. Diagram vysvetlite.

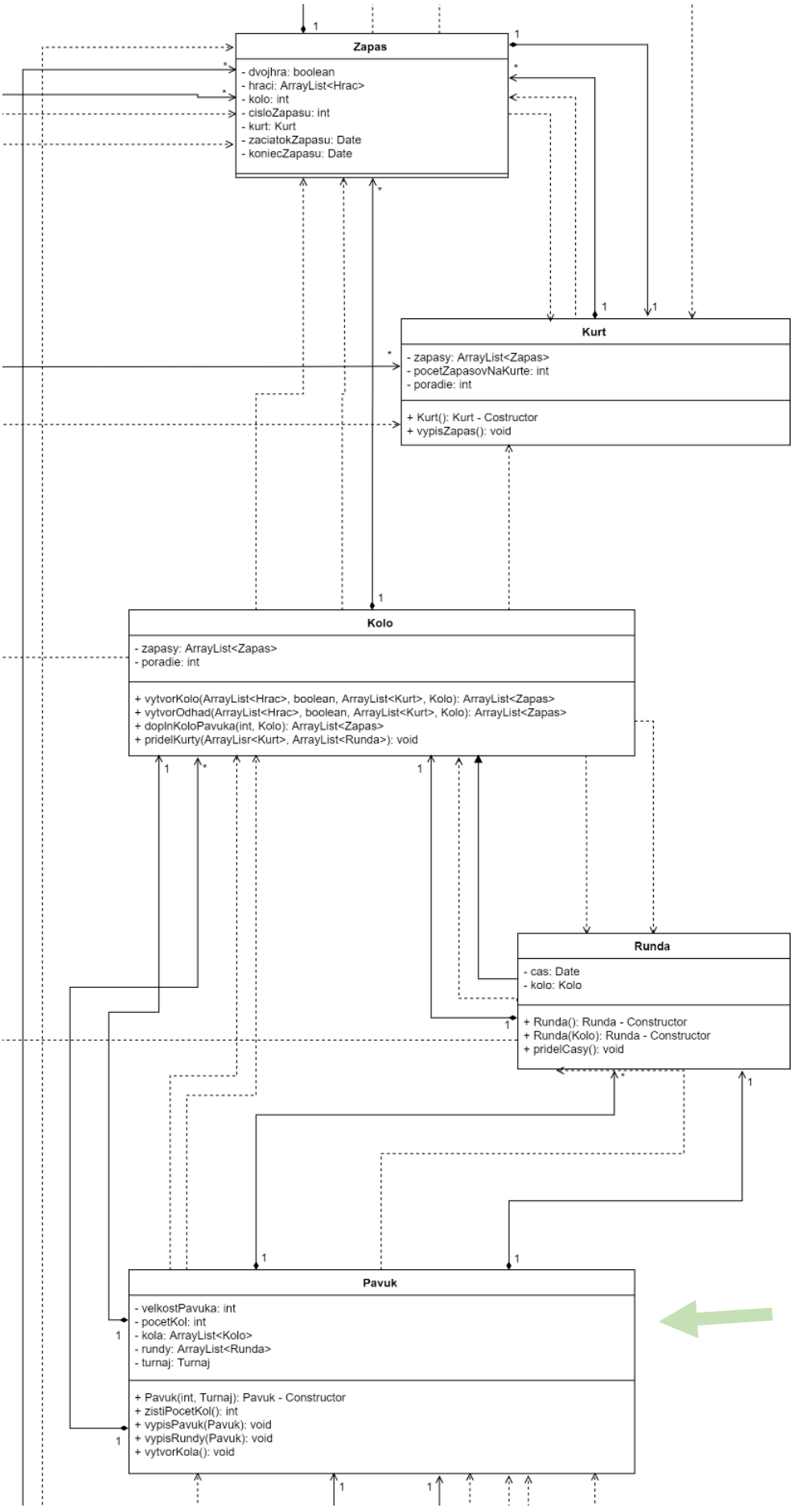



```java
public interface Zapasy {

    ArrayList<Hrac> getHraci();

    void setHraci(ArrayList<Hrac> hraci);

    void setZaciatokZapasu(Date zaciatokZapasu);
    int getKolo();
    void setKolo(int kolo);
    Kurt getCobain();
    void setCobain(Kurt cobain);
    boolean getDvojhra();
    void setDvojhra(boolean dvojhra);

}
```

Metódy predpísané v interface Zapasy sa implementujú v classe Zapas, používajú sa napríklad v triede Pavúk : `for (int i2 = 0; i2 < kolo.getZapasy().get(i1).getHraci().size(); i2++) {`

```java
public class Zapas implements Serializable, Zapasy {
    private boolean dvojhra;
    private ArrayList<Hrac> hraci;
    ....

    public ArrayList<Hrac> getHraci() {
        return hraci;
    }
```

Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová



Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

3. (7 b) Na príklade z vlastného riešenia zadania vysvetlite rozdiel medzi uplatnením a prejavom polymorfizmu. V prípade ak ste ho neimplemntovali navrhnite riešenie teraz. Zároveň zdôvodnite vhodnosť jeho použitia v kontexte vlastného riešenia.

Je to trieda, ktorá prekonáva metódu ktorú dedí – objekt má schopnosť nadobúdať viacero „foriem“

Prekonávanie je spôsob akým vieme implementovať metódu ktorá už bola implementovaná v rodičovskej triede – rovnaká metóda pro jednotlivých objektoch robí niečo iné

class MajstrovstvaSR – dedí z triedy TurnajA, ktorá dedí z triedy Turnaj

```
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(velkostPavuka);
    Pavuk pavuk;
    MajstrovstvaSR MS = new MajstrovstvaSR(3, true, velkostPavuka, datum,
hrac);
    pavuk = new Pavuk(velkostPavuka, MS);
    MS.setPavuk(pavuk);
    return MS;
}
```

abstract class Turnaj

```
public static Turnaj vytvorTurnaj() {

        .....
    Date datum = new Date();
    System.out.println("Zadaj cislo dna v mesiaci:");
    datum.setDate(keyboard.nextInt());
    System.out.println("Zadaj hodinu zaciatku turnaja:");
    datum.setHours(keyboard.nextInt());
    System.out.println("Zadaj minuty:");
    datum.setMinutes(keyboard.nextInt());
    datum.setSeconds(0);

    return turnaj;
}
```

Prekonáva zdedenú metódu vytvorTurnaj, preto že pri triede MajstrovstvaSR napríklad vieme dopredu dátum uskutočnenia sa, a netreba ho získavať zo vstupu používateľa.

Uplatnenie = reálna zmena funkcionality (nepotrebnosť vstupu používatela)

Prejav = viditelný output, výpis... (fixný dátum)

Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

4. (7 b) Vo vašom projekte ste možno implementovali rolu pozorovateľa, ktorý bol v prípade potreby informovaný o jasne definovanej situácii na strane pozorovaného objektu. Uveďte dotknutú časť kódu (v prípade ak ste to neimplemntovali navrhnite riešenie teraz). Poukážte na dôležité súčasti. V hierarchii pozorovateľa použite rozhranie aj abstraktnú triedu.

Observer je trieda používaná na sledovanie stavu nejakej tiedy – keď sa stav zmení, trieda upovedomí observer, a ten si stav zaznamená.

```
18  public abstract class Turnaj extends Turnaje implements Serializable {
19      protected static int velkostPavuka;
20      protected Kategoria kategoria;
21      protected int pocetKurtov;
22      protected int pocetDni;
23      protected int zisk = 0;
24      protected Date zaciatokTurnaja;
25      protected ArrayList<Hrac> hraci;
26      protected ArrayList<Zapas> zapasy;
27      protected ArrayList<Den> dni;
28      protected ArrayList<Kurt> zoznamKurtov;
29      protected Pavuk pavuk;
30      protected PrvyZapasPozorovatel pozorovatel;
31
32      public void obnovPozorovatela(){
33          pozorovatel.update(zapasy.get(0));
34      }
35
36      public void vytvorPozorovatela(){
37          this.pozorovatel = new PrvyZapasPozorovatel();
38      }
39
```

Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

Trieda PrvyZapasPozorovatel rozširuje triedu Pozorovatel, kt. implementuje rozhranie Pozorovatelia.

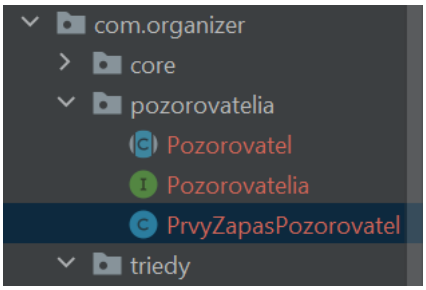
Metóda update z rozhrania sa implementuje v triede PrvyZapasPozorovatel a používa sa v triede Turnaj.

```
package com.organizer.pozorovatelia;

public abstract class Pozorovatel implements Pozorovatelia {

}
```

```
package com.organizer.pozorovatelia;

import com.organizer.triedy.Zapas;

public interface Pozorovatelia {
    public void update(Zapas data);
}
```

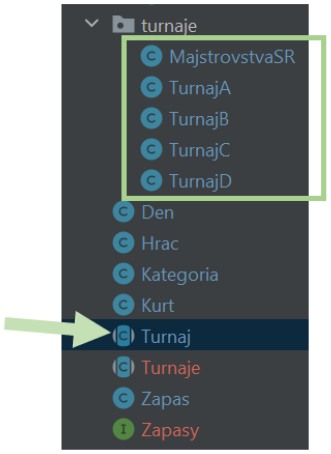
```
package com.organizer.pozorovatelia;

import com.organizer.triedy.Zapas;

public class PrvyZapasPozorovatel extends Pozorovatel{
    private Zapas posledny;

    @Override
    public void update(Zapas data) {
        this.posledny = data;
    }
}
```

Veci súvisiace s pozorovatelom sa nachádzajú v balíku pozorovatelia



Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová

5. (7 b) Uveďte príklad s vysvetlením z vlastného riešenia v zadaní, kde pridávate (v prípade ak ste to neimplemntovali, kde by ste pridali) funkcionalitu objektom istej triedy bez toho aby ste museli do nich zasahovať. Dotknuté triedy vyznačte aj diagrame UML vrátane relevantných vzťahov.



Kebyže chceme pridať metódu do niektorého z turnajov (TurnajA...), pridáme ju do classu Turnaj, preto že „turnaje“ z neho dedia.

Napríklad v class Turnaj :

```
public Pavuk getPavuk() {
    return pavuk;
}

public void setPavuk(Pavuk pavuk) {
    this.pavuk = pavuk;
}

public ArrayList<Hrac> getHraci() {
    return hraci;
}

public void setHraci(ArrayList<Hrac> hraci) {
    this.hraci = hraci;
}

public ArrayList<Zapas> getZapasy() {
    return zapasy;
}

public void setZapasy(ArrayList<Zapas> zapasy) {
    this.zapasy = zapasy;
}

public ArrayList<Den> getDni() {
    return dni;
}

public void setDni(ArrayList<Den> dni) {
    this.dni = dni;
}

public ArrayList<Kurt> getZoznamKurtov() {
    return zoznamKurtov;
}
```

Základy objektovo-orientovaného programovania
2020/2021 Ing. Ján Lang, PhD., ÚISI FIIT STU Skúška - 07. január 2021, 13:30 - 15:30
Emma Macháčová