18.Projekt Škola

18.1. Základní popis, zadání úkolu

Pracujeme na projektu Škola, který je ke stažení na `java.vse.cz`. Je to pouze fragment aplikace – cílem je ukázat, jak v instancích zachytit stromovou strukturu. Po dokončení inicializace aplikace se vytvoří instance třídy `Skola`. Po zavolání metody `vypis()` by se na obrazovku měla vypsat organizační struktura školy (variantně s pracovníky či bez pracovníků).
V tomto projektu budeme řešit tyto úkoly:

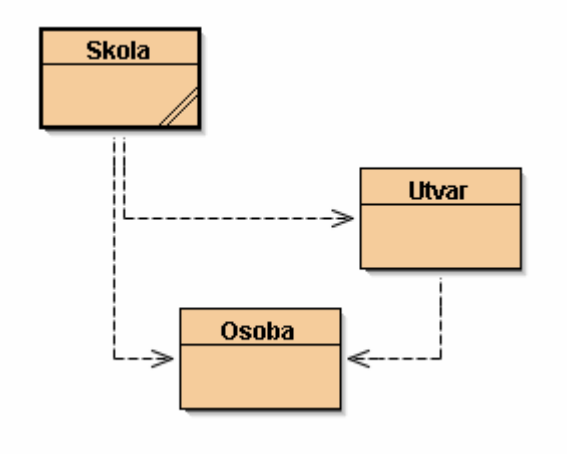
- navrhnout datové atributy ve třídě `Utvar` pro uložení podřízených útvarů a osob pracujících v útvaru,
- doplnit obsah metod `pridej()` ve třídě `Utvar`,
- vypsat seznamu podřízených útvarů, tj. doplnit obsah metod `vypis()` ve třídě `Utvar`.

Tento projekt má následující cíle:

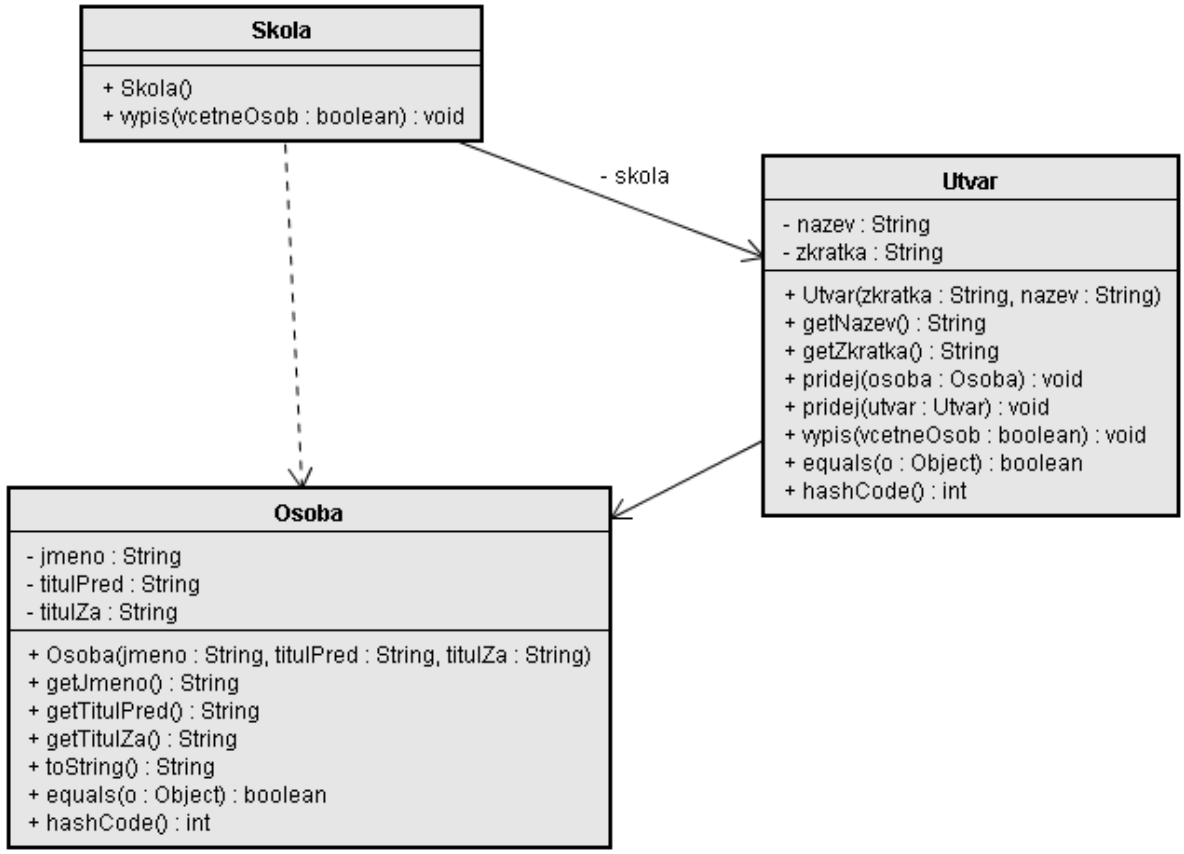
- ukázat vytvoření složitější datové struktury – v tomto projektu se vytvoří stromová struktura instancí (organizační struktura školy). Projekt též ukazuje rozdíly mezi strukturou tříd a strukturou instancí.
- ukázat přetěžování metod na metodě `pridej()`.

18.2. Struktura tříd

Projekt Škola se skládá s následujících tříd (obrázek z BlueJ):



Obrázek 18.1 Diagram tříd projektu Škola převzatý z BlueJ



Obrázek 18.2 Diagram tříd projektu Škola včetně datových atributů a metod

18.3. Popis kódu třídy Skola

Třída `Skola` je hlavní třídou, ve které se vytvářejí zkušební data. Třída dále obsahuje metodu `vypis()`, přes kterou se vypisuje organizační struktura. V konstruktoru třídy `Skola` se vytvářejí jednotlivé testovací instance třídy `Utvar` a `Osoba` a vzájemně se spojují mezi sebou.

```
 1 /**
 2  * Tato třída je součástí projektu Skola a obsahuje základní
 3  * třídu pro školu. V konstruktoru je i inicializace
 4  * organizační struktury školy.
 5  *
 6  * @author     Luboš Pavlíček
 7  * @version    1.0, srpen 2004
 8  */
 9 public class Skola {
10
11   private Utvar skola;
12   /**
13    * Konstruktor pro instance třídy Skola. Vytvářejí se
14    * instance tříd Utvar a Osoba, které se propojují do sebe.
15    */
16   public Skola() {
17      skola = new Utvar("VŠE","Vysoká škola ekonomická");
18      Utvar fak1 = new Utvar("F1","Fakulta financí a účetnictví");
19      Utvar fak2 = new Utvar("F2","Fakulta mezinárodních vztahů");
20      Utvar fak3 = new Utvar("F3","Fakulta podnikohospodářská");
```

```
21      Utvar fak4 = new Utvar("F4",
22              "Fakulta informatiky a statistiky");
23      Utvar fak5 = new Utvar("F5","Fakulta národohospodářská");
24      Utvar fak6 = new Utvar("F6","Fakulta managementu");
25      skola.pridej(fak1);
26      skola.pridej(fak2);
27      skola.pridej(fak3);
28      skola.pridej(fak4);
29      skola.pridej(fak5);
30      skola.pridej(fak6);
31      fak4.pridej(new Utvar("KMAT","katedra matematiky"));
32      Utvar kstp = new Utvar("KSTP","katedra statistiky
                a pravděpodobnosti");
33      fak4.pridej(new Utvar("KEST",
34              "katedra ekonomické statistiky"));
35      fak4.pridej(new Utvar("KDEM","katedra demografie"));
36      fak4.pridej(new Utvar("KEKO","katedra ekonometrie"));
37      fak4.pridej(new Utvar("KSA","katedra systémové analýzy"));
38      fak4.pridej(new Utvar("KIZI",
39              "katedra informačního a znalostního inženýrství"));
40      fak4.pridej(new Utvar("KFIL","katedra filosofie"));
41      Utvar kit = new Utvar("KIT",
42              "katedra informačních technologii");
43      fak4.pridej(kit);
44      fak4.pridej(kstp);
45      skola.pridej(new Osoba("Durčáková Jaroslava", "Doc. Ing.",
46                  "CSc."));
47      Osoba dekan = new Osoba ("Hindls Richard", "Prof. Ing.",
48                  "CSc.");
49      fak4.pridej(dekan);
50      kstp.pridej(dekan);
51      kit.pridej(new Osoba("Vořišek Jiří","Prof. Ing.", "CSc."));
52      kit.pridej(new Osoba("Buchalcevová Alena", "Ing.","PhD."));
53      kit.pridej(new Osoba("Pavlíčková Jarmila", "Ing.",""));
54      kit.pridej(new Osoba("Pavlíček Luboš","Ing.",""));
55      kit.pridej(new Osoba("Tichý Vladimír", "RNDr.",""));
56    }
57    /**
58     * Metoda vypíše organizační strukturu
59     *
60     * @param  vcetneOsob   zda vypsat i osoby přiřazené do
61     *                      jednotlivých útvarů
62     */
63    public void vypis(boolean vcetneOsob) {
64    skola.vypis(vcetneOsob);
65    }
66  }
```

18.4. Popis kódu tříd Utvar a Osoba

Nebudeme zde uvádět kód tříd *Utvar* a *Osoba*, neboť jsou jednoduché. Třída *Osoba* popisující jednu osobu má tři datové atributy – *jmeno*, *titulPred* a *titulZa*, které se inicializují přes konstruktor. Třída *Osoba* dále obsahuje metody pro získání hodnot těchto atributů a metody *toString()*, *equals()* a *hashCode()*.

Třída *Utvar*, která představuje jeden útvar z organizační struktury má dva datové atributy – *nazev* a *zkratka*, které se inicializují přes konstruktor. Vedle metod pro získání těchto datových atributů má

metody `equals()` a `hashCode()`. Dále třída obsahuje hlavičky metod `pridej()` pro vložení podřízených útvarů a pracovníků útvaru a hlavičku metody `vypis()` pro vypsání označení útvaru, pracovníků útvaru a podřízených útvarů. Úkolem je doplnit do této třídy datové atributy pro uložení podřízených útvarů a pracovníků útvaru a dokončit metody `pridej()` a `vypis()`.

18.5. Postup řešení

18.5.1. Deklarace datových atributů a vkládání instancí

Pro uložení podřízených útvarů můžeme použít:

- seznam (List),
- množinu (Set),
- mapu (Map),
- pole (Array).

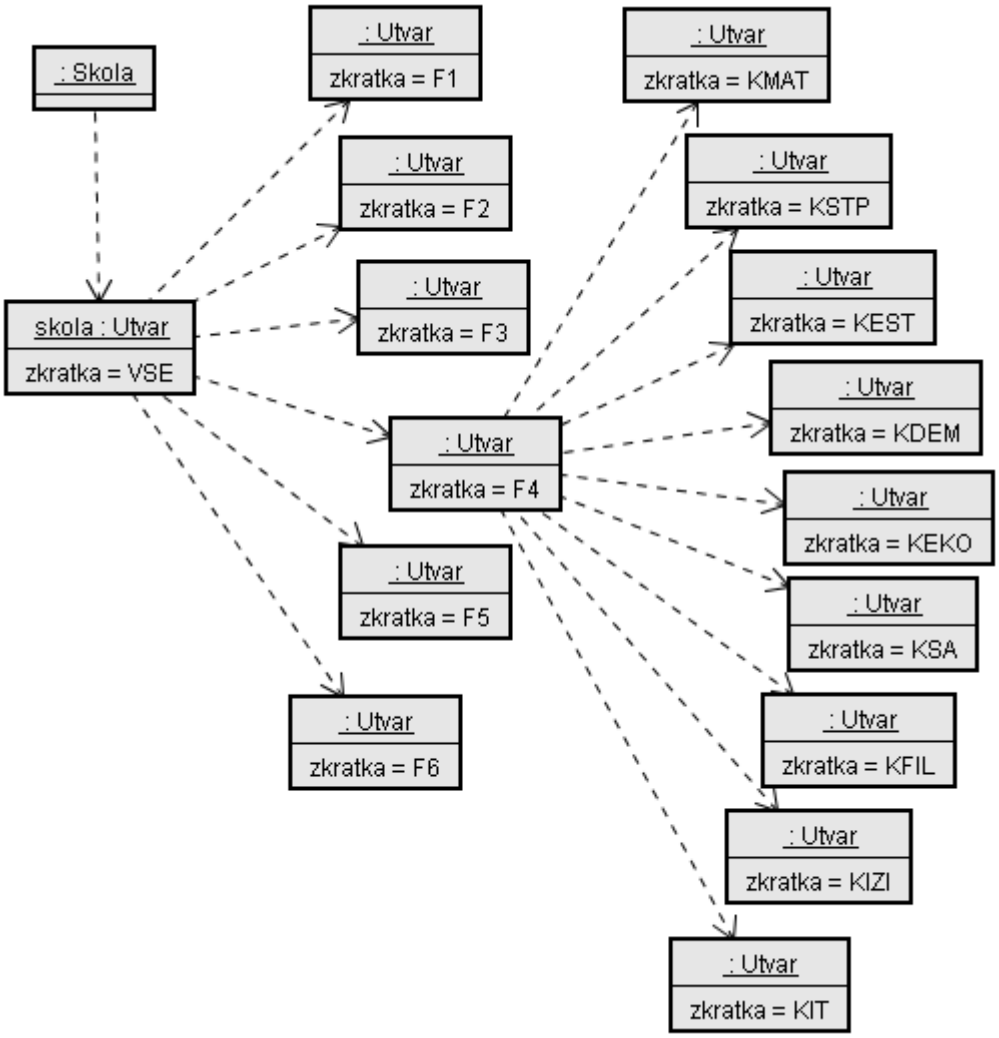
Vzhledem k tomu, že podřízené útvary by měly být jedinečné, odpovídá nejlépe množina. Je splněna i podmínka pro použití množin – třída `Utvar` má implementovány metody `equals()` a `hashCode()`. Z implementací množiny použijeme nejrychlejší – `HashSet`. V deklaraci nadefinujeme datový atribut jako typ `Collection`, neboť nepotřebujeme používat speciální metody množiny a použití tohoto typu nám umožňuje v budoucnosti snadnou změnu implementace.
Pro ukládání pracovníků útvaru též použijeme množinu (třída `Osoba` má též implementovány metody `equals()` a `hashCode()`).
Následuje deklarace datových atributů a konstruktor třídy `Utvar`:

```
    private String nazev;
    private String zkratka;
    private Collection <Utvar> podrizene;
    private Collection <Osoba> pracovnici;

    public Utvar(String zkratka, String nazev) {
         this.zkratka = zkratka;
         this.nazev = nazev;
         podrizene = new HashSet <Utvar>();
         pracovnici = new HashSet <Osoba>();
    }
```

Pro vkládání podřízených útvarů a pracovníků útvaru se doplní předpřipravené metody `pridej()`:

```
    public void pridej (Osoba osoba) {
         pracovnici.add(osoba);
    }

    public void pridej (Utvar utvar) {
         podrizene.add(utvar);
    }
```

Všimněte si, že jsou zde dvě metody stejného jména s různým typem parametrů. Jedná se o tzv. přetížení metod. Volba správné metody se provádí na základě typu parametru. Podívejte se na kód konstruktoru třídy `Skola` – zde to vypadá, že se používá stejná metoda pro přidání jak podřízených útvarů, tak pracovníků útvaru. Přetěžování metod zjednodušuje psaní kódu na straně používání metod i částečně na straně třídy, která metodu poskytuje (přijímá zprávy).
Následující diagram zobrazuje vztahy mezi instancí třídy `Skola` a instancemi třídy `Utvar`.
Z diagramu je zřejmé, že jsme vytvořili stromovou strukturu instancí. Z důvodů přehlednosti nejsou do diagramu začleněny instance třídy `Osoba` – jejich doplnění by pro čtenáře neměl být problém (nezapomeňte, že na instanci Prof. Hindlse budou dva odkazy – jednou z fakulty F4, podruhé z útvaru KSTP).



Obrázek 18.3 Diagram zobrazuje vztahy mezi instancí třídy Skola a instancemi třídy Utvar

18.5.2. Výpis organizační struktury

V první verzi metody *`vypis()`* ve třídě *`Utvar`* se nejdříve vypíše zkratka a jméno útvaru a poté se postupně volá metoda *`vypis()`* pro všechny podřízené útvary (aby se vypsala jména všech podřízených útvarů). Pokud podřízený útvar má opět podřízené útvary (fakulta má katedry), tak se opět vypisují jejich jména.

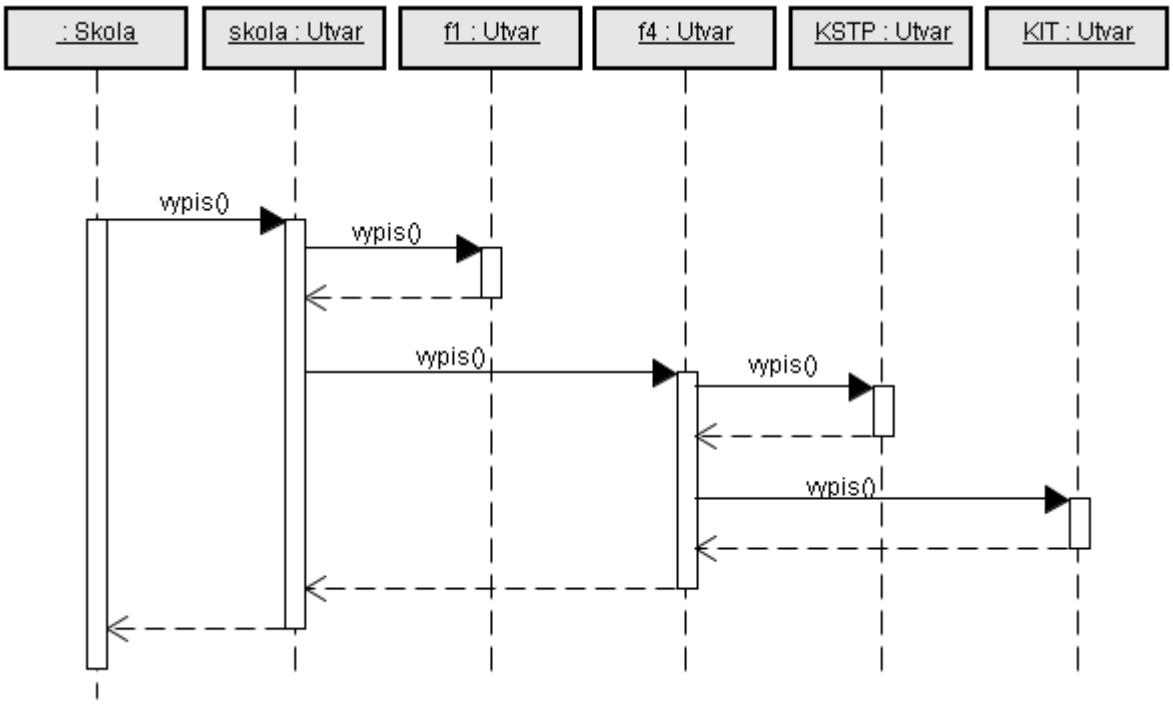
```
    public void vypis(boolean vcetneOsob) {
         System.out.println(zkratka+" - " + nazev);
         for (Utvar utvar : podrizene) {
              utvar.vypis(vcetneOsob);
         }
    }
```

```
VŠE - Vysoká škola ekonomická
F3 - Fakulta podnikohospodářská
F5 - Fakulta národohospodářská
F6 - Fakulta managementu
Fl - Fakulta financí a účetnictví
F4 - Fakulta informatiky a statistiky
KIZI - katedra informačního a znalostního inženýrství
KEST - katedra ekonomické statistiky
KIT - katedra informačních technologii
KDEM - katedra demografie
KEKO - katedra ekonometrie
KFIL - katedra filosofie
KSTP - katedra statistiky
KSA - katedra systémové analýzy
KMAT - katedra matematiky
F2 - Fakulta mezinárodních vztahů
```

Obrázek 18.4 Výpis organizační struktury pomocí první verze metody vypis()

Když si nyní vytvoříme instanci třídy `Skola` a zavoláme metodu `vypis()`, bude vypsaná organizační struktura vypadat podobně jako na obrázku 18.4. Výsledek má dva nedostatky:

- útvary se nevypisují setříděně (nejlépe dle abecedy),
- nejsou vidět úrovně organizační struktury – je potřeba vhodně doplnit odsazování.



Obrázek 18.5 Schématické znázornění volání metod vypis() u jednotlivých instancí v situaci, kdy by v aplikaci byly dvě fakulty a dvě katedry

18.5.3. Setřídění výpisu

Problém se tříděním lze vyřešit několika způsoby. My si zde popíšeme nejjednodušší – pro uložení instancí podřízených útvarů se použije třída `TreeSet()`, která automaticky třídí uložené instance.

Předpokladem pro použití této třídy je však implementace rozhraní `Comparable` u objektů, které se mají vkládat do této datové struktury. Hlavičku třídy `Utvar` je potřeba proto rozšířit o implementaci rozhraní:

```
public class Utvar implements Comparable <Utvar> {
```

Rozhraní `Comparable` vyžaduje, aby třída obsahovala metodu

```
    public int compareTo(Utvar druhy)
```

která vrací

- kladnou hodnotu, pokud aktuální objekt má být zařazen před objekt zadaný jako parametr,
- nulu, pokud na pořadí nezáleží (tj. je jedno, v jakém pořadí budou),
- zápornou hodnotu, pokud aktuální objekt má být zařazen za objekt zadaný jako parametr.

Metoda `compareTo()` by mohla vypadat následovně[38]:

```
    public int compareTo(Utvar druhy) {
        return zkratka.compareTo(druhy.zkratka);
    }
```

Třídit se bude dle zkratky – využije se metody `compareTo()` ve třídě `String`.
Vlastní změna z dynamické struktury `HashSet` na `TreeSet` je triviální úprava v konstruktoru třídy `Utvar` (nezapomeňte doplnit `import java.util.TreeSet`):

```
    podrizene = new TreeSet <Utvar>();
```

Pro setřídění osob v útvaru je potřeba též implementovat rozhraní `Comparable` ve třídě `Osoba` a u datového atributu `pracovnici` použít implementaci `TreeSet`.
Pokud by jste ukládali útvary či osoby do seznamu (List), tak je možné tento seznam setřídit pomocí

```
    Collections.sort(pracovniciList);
```

Opětně je předpokladem, že objekty vložené do seznamu implementují rozhraní `Comparable`.
Statická metoda `Collections.sort()` nemůže třídit množiny (implementace Set).

18.5.4. Zvýraznění organizační struktury (odsazování)

Odsazování je složitější – abychom věděli, o kolik by se měl název konkrétního útvaru odsunout, potřebujeme vědět, na jaké úrovni řízení příslušný útvar je (na nulté úrovni je celá škola, na první úrovni fakulty, na druhé úrovni jsou katedry). Ve třídě `Utvar` si však neuchováváme informaci o této úrovni, ani o nadřízeném útvaru (pokud by byl známý nadřízený útvar, bylo by možné úroveň dopočítat). Úroveň řízení lze též zjistit z průběhu vlastního výpisu – začíná se na nulté úrovni, při výpisu podřízených útvarů se úroveň řízení zvyšuje. Doplníme tedy metodu `vypis()` o druhý parametr, který bude určovat úroveň řízení.
Metoda `vypis()` by nyní mohla vypadat následovně:

```
    public void vypis(boolean vcetneOsob, int uroven) {
        for (int i=0; i< uroven; i++) {
            System.out.print("   ");
        }
        System.out.println(zkratka+" - " + nazev);
        for (Utvar utvar : podrizene) {
            utvar.vypis(vcetneOsob, uroven+1);
        }
    }
```

[38] Někoho by mohlo na tomto kódu překvapit, že se porovná privátní datový atribut `zkratka` z této instance s privátním datovým atributem `zkratka` z druhé instance – vypadá to na porušení zapouzdření. Modifikátory přístupu ale určují přístup k datovým atributům a metodám na úrovni třídy, ne na úrovni instancí. Zde jsou obě proměnné ze stejné třídy, tudíž k narušení zapouzdření u privátních proměnných nedochází.

Před spuštěním ještě nezapomeňte upravit volání metody *vypis()* ve třídě *Skola* – je potřeba doplnit druhý parametr. Výpis poté bude vypadat následovně:

```
VŠE - Vysoká škola ekonomická
   F1 - Fakulta financí a účetnictví
   F2 - Fakulta mezinárodních vztahů
   F3 - Fakulta podnikohospodářská
   F4 - Fakulta informatiky a statistiky
      KDEM - katedra demografie
      KEKO - katedra ekonometrie
      KEST - katedra ekonomické statistiky
      KFIL - katedra filosofie
      KIT - katedra informačních technologii
      KIZI - katedra informačního a znalostního inženýrství
      KMAT - katedra matematiky
      KSA - katedra systémové analýzy
      KSTP - katedra statistiky
   F5 - Fakulta národohospodářská
   F6 - Fakulta managementu
```

Obrázek 18.6 Výpis útvarů se setříděním a odsazením jednotlivých úrovní

18.5.5. Doplnění výpisu osob

Vypsání pracovníků je jednoduché – do metody *vypis()* se doplní cyklus procházející seznam pracovníků. Jména pracovníků je též vhodné odsazovat – protože odsazování je nyní potřeba na více místech, přesuneme kód pro odsazování do samostatné privátní metody. Metody *vypis()* a *odsadit()* by mohly vypadat následovně:

```
    public void vypis(boolean vcetneOsob, int uroven) {
         odsadit(uroven*3);
         System.out.println(zkratka+" - " + nazev);
         if (vcetneOsob) {
              for (Osoba osoba : pracovnici) {
                   odsadit((uroven+1)*3);
                   System.out.println("- " + osoba.toString());
              }
         }
         for (Utvar utvar : podrizene) {
              utvar.vypis(vcetneOsob, uroven+1);
         }
    }

    private void odsadit (int pocetMezer) {
         for (int i=0; i<pocetMezer; i++) {
              System.out.print(" ");
         }
    }
```

18.6. Domácí úkoly

1. Doplňte do třídy *Skola* metodu, která by měla jako parametr zkratku útvaru a tento útvar vyhledala a vypsala.

2. Rozšiřte předchozí zadání o vypsání i nadřízených útvarů, tj. pokud bude parametrem „KIT“, vypíše se
   VSE – Vysoká škola ekonomická
   F4 – Fakulta informatiky a statistiky
   KIT – katedra informačních technologií

3. Vytvořte obdobnou metodu, jako je v předchozím zadání, pro vyhledání osob. Nezapomeňte, že někteří pracovníci jsou ve více útvarech.

4. Doplňte evidenci funkcí u osob – např. rektor, děkan, vedoucí katedry, člen katedry. Zamyslete se nad tím, zda stačí některou třídu doplnit o datové atributy či je potřeba použít další třídu. Nezapomeňte, že některé osoby mohou mít různé funkce v různých útvarech (např. děkan je současně i vedoucím/členem katedry).

5. Rozšiřte předchozí úlohu o setřídění výstupu dle funkcí – např. u katedry by se osoby měly vypisovat dle následujícího pořadí funkcí:
   vedoucí katedry,
   zástupce vedoucího katedry,
   sekretářka,
   profesoři,
   docenti,
   odborní asistenti,
   asistenti,
   doktorandi.