# FAKULTA ELEKTROTECHNIKY A INFORMATIKY

## DATOVÉ STRUKTURY

### ZADÁNÍ SEMESTRÁLNÍ PRÁCE B

### KORPORÁTNÍ FIRMA

Maximální možný bodový zisk: **4 body**

**A) Motivační příklad:**

Korporátní firma disponuje celou řadou poboček. Každá pobočka má hierarchicky uspořádané pracovní pozice, do kterých zařazuje zaměstnance.

**B) Použité datové struktury:**

Primární datová struktura bude umožňovat vyhledávání jednotlivých poboček nebo zaměstnanců. Struktura bude implementována v samostatném třídě `AbstrTable` jako abstraktní datová struktura tabulka. Třída `AbstrTable` bude později adaptována na konkrétní použití. V našem případě se bude jednat o třídy `Firma` a `Zamestnanci`. Implementace třídy `AbstrTable` bude provedena pomocí modifikované datové struktury `AbstrDoubleList` ze semestrální práce A.
Třída `AbstrTable` bude navržena s typovými parametry `<K, V>`. Kde `K` bude klíč do tabulky a `V` bude datová entita v tabulce a dále budou implementovány následující metody rozhraní:

| | |
|---|---|
| `void zrus()` | zruší celou tabulky |
| `boolean jePrazdny()` | zjistí, zda tabulka je prázdná |
| `V najdi(K key)` | vyhledá prvek dle klíče |
| `void vloz(K key, V value)` | vloží prvek s klíčem do tabulky |
| `V odeber(K key)` | odebere prvek dle klíče z tabulky |
| `Iterator<V> iterator()` | vytvoří iterátor, který umožňuje procházení tabulky |

Sekundární datovou strukturou bude organizační struktura pobočky. Hierarchická stromová struktura bude implementována v samostatné třídě `AbstrTree`, v které bude realizována funkce neuspořádaného kořenového (k-cestného) stromu (NKS). Třída `AbstrTree` bude později adaptována na konkrétní použití. V našem případě se bude jednat o třídu `Pobocka`. Třída `AbstrTree` bude využívat k vedení odkazů na podřízené uzly třídu `AbstrDoubleList` ze semestrální práce A.

Třída `AbstrTree` bude parametrizována typovým parametrem `E`. Typový parametr bude zastupovat skutečnou datovou entitu uzlu stromu. Třída `AbstrTree` bude implementovat následující metody rozhraní:

| | |
|---|---|
| `void zrus()` | zruší celý strom |
| `boolean jePrazdny()` | zjistí, zda strom je prázdný |
| `int mohutnost()` | dodá počet prvků stromu |
| `void vlozKoren(E data)` | vloží „inicializační“ uzel (kořen) stromu |
| `void vlozList(E data)` | vloží další list stromu jako syna aktivního prvku |
| `E odeberKoren()` | odebere kořen (pouze když obsahuje jen kořen) |
| `E odeberList(int poradi)` | odebere list aktivního uzlu, který je dán pořadím |
| `E zpristupniKoren()` | zpřístupní kořen stromu |
| `E zpristupniSyna(int poradi)` | zpřístupní syna aktivního uzlu, který je dán pořadím |
| `E zpristupniOtce()` | zpřístupní otce aktivního uzlu |
| `Iterator<E> iterator()` | vytvoří iterátor procházení stromu do hloubky |

**Poznámky:**

1. Pořadí synů bude číslováno od hodnoty 1.
2. Pouze metody typu „zpřístupni“ budou nastavovat aktuální uzel stromu.
3. Metody typu „vlož“ nebudou neměnit aktuální uzel.
4. Iterátor bude využívat ADS zásobník nebo frontu (podle potřeby), který/á bude postaven/a nad ADS ze semestrální práce A
5. V identifikátorech je zakázáno používat české diakritické znaky.

**C) Pro ověření funkčnosti implementovaných ADS budou vytvořeny třídy**

**Třída** `Firma`
Třída bude implementovat rozhraní s operacemi vyhledání, vložení a odebrání pobočky ze seznamu. Třída bude též poskytovat datovod (stream), který může být využit k procházení seznamu poboček při konstrukci uživatelského rozhraní. Tato třída bude vlastně adaptérem na třídu `AbstrTable`.

Třída bude implementovat rozhraní s těmito operacemi:

| | |
|---|---|
| `Pobocka najdi(String nazev)` | najdi pobočku podle jejího jména |
| `vloz(Pobocka)` | vlož pobočku do seznamu poboček podle jména pobočky |
| `Pobocka odeber(String nazev)` | odeber pobočku ze seznamu |

**Třída** `Pobocka`
Tato třída bude udržovat hierarchickou stromovou strukturu pracovních pozic v pobočce. Třída bude adaptovat třídu `AbstrTree`, která bude parametrizována třídou `Pozice`.

Pracovní pozice bude základním organizačním prvkem hierarchie pobočky. Konkrétními pozicemi můžou být ředitel pobočky, vedoucí úseku, vedoucí oddělení a pracovník oddělení. Ředitel pobočky bude kořenem a pracovník oddělení bude listem ve stromové struktuře. Ke každé pozici v organizačním schématu pobočky bude možné přiřadit konkrétního pracovníka ze seznamu zaměstnanců.

Pobočka bude dále udržovat informace o svém názvu, o počtu pracovních pozic a místě umístění pobočky (město). Třída bude též poskytovat datovod (stream), který může být využit k procházení seznamu pozic pobočky.

Třída `Pobocka` bude implementovat rozhraní, které bude mít především tyto operace:

| | |
|---|---|
| `Pozice zpristupniReditele()` | zpřístupni ředitele pobočky |
| `Pozice zpristupniPodrizenouPozici(int n)` | zpřístupni n-tou podřízenou pozici aktuální (nadřazené) pozice |
| `Pozice zpristupniNadrizenouPozici()` | zpřístupni nadřízenou pozici aktuální pozice |
| `vlozPozici(Pozice)` | vloží novou pozici jako list k aktuální pozici |
| `Pozice odeberPozici(int n)` | odebere n-tou pozici za předpokladu, že nemá podřízené |
| `Iterator<Pozice> iterator()` | vytvoří iterátor procházení pozic (do hloubky) |

**Třídy** `Zamestnanci` **a** `Zamestnanec`
Třída `Zamestnanci` bude udržovat seznam všech zaměstnanců korporátní firmy. Zaměstnanci se budou přiřazovat do pozic na pobočkách. Třída bude vystavěna na třídě `AbstrTable`. Pro každého zaměstnance se bude evidovat jeho osobní číslo, jméno a příjmení. Přístup k jednotlivým zaměstnancům seznamu bude zajišťován jejich evidenčním číslem.
Třída `Zamestnanec` bude obsahovat informace s osobními údaji jako je jméno, příjmení, e-mail a osobní číslo.

**Třídu** `Pozice`
Při implementaci třídy `Pozice` bude možné použít jednu z těchto možností:

1. Třída `Pozice` bude abstraktní třídou a proto bude předkem specializovaných tříd jako jsou třídy `ReditelPobocky`, `VedouciUseku`, `VedoucíOddeleni` a `PracovnikOddeleni`.
2. Třída `Pozice` nebude abstraktní třídou a nebude mít potomky. Bude ale obsahovat informaci o typu pozice.

K pozici bude možné přiřadit zaměstnance ze seznamu zaměstnanců. Třídy `Pozice` a `Zamestnanec` potom budou tvořit jeden celek pro výpis organizačního schématu pobočky. Je povoleno jednoho zaměstnance přiřadit na více pracovních pozic.

**Generátor**
Pro ověření třídy `Firma` bude k dispozici samostatný generátor zkušebních dat, který naplní nejméně čtyři pobočky, pro každou pobočku vytvoří jedinečné organizační schéma a do jednotlivých pozic na pobočce přiřadí konkrétního pracovníka ze seznamu zaměstnanců. Tento generátor musí vyhovovat návrhovému vzoru Library. Ukázka takového generátoru je v příloze tohoto zadání.

**D) Uživatelské rozhraní** Aplikace `KorporatniFirma` bude využívat uživatelské formulářové rozhraní v JavaFX, kterým bude možné ověřit operace tříd `Firma`, `Pobocka` a `Zamestnaci`.

**E) Ostatní požadavky**

1. Aplikace `KorporatniFirma` nechť dále umožňuje zadávání vstupních dat z klávesnice, ze souboru, výstupy z programu nechť je možné zobrazit na obrazovce a uložit do souboru.
2. Požaduje se, k první konzultaci vypracovat diagram tříd s návrhem aplikace `KorporatniFirma`.
3. Při odevzdání semestrální práce se, kromě projektu, požaduje soubor s ilustračním příkladem (i grafické formě) korporátní firmy, sestávající nejméně ze 4 poboček a 3 hierarchických úrovní na pobočku, který bude uložen v projektu aplikace.

4. Všechny chybové stavy tříd budou ošetřeny výjimkami, které se budou zachycovat a zobrazovat až v uživatelském rozhraní aplikace.
5. Třídy `AbstTable` a `AbstrTree` musí být ověřeny jednotkovými testy v JUNIT.
6. Aplikace `KorporatniFirma` musí být pro platformu Java 8.
7. Název projektu v NetBeans musí začínat příjmením studenta.

# Příloha

Příklad generátoru dat.

```
package korporatnifirma;
import lorem.Lorem;
import lorem.LoremIpsum;// https://github.com/mdeanda/lorem
import static korporatnifirma.TypPozice.*;

public final class GeneratorDat {
  private static final Lorem lorem = new LoremIpsum();
  private static int indexZamestnanec = 1;
  private GeneratorDat() {}
  public static IZamestnanci generujZamestnance() {
    IZamestnanci zamestnanci = new Zamestnanci();
      for (int i = 1; i < 600; i++) {
        zamestnanci.vloz(i,
        new Zamestnanec(i, lorem.getFirstName(), lorem.getLastName(), lorem.getEmail()));
      }
    return zamestnanci;
  }
  public static IPobocka generujPobocku(IZamestnanci zamestnanci)
        throws IllegalAccessException {
    Pobocka pobocka = new Pobocka(lorem.getWords(1), lorem.getCity());
    Pozice pozice = new Pozice(REDITEL, zamestnanci.najdi(indexZamestnanec++));
    pobocka.vlozPoziciReditele(pozice);
    pobocka.zpristupniReditele();
    int pocetUseku = nahodneCislo(1, 2);
    for (int i = 0; i < pocetUseku; i++) {
       pozice = new Pozice(VEDOUCI_USEKU, zamestnanci.najdi(indexZamestnanec++));
       pobocka.vlozPozici(pozice);
       pobocka.zpristupniPodrizenouPozici(i + 1);
       int pocetOddeleni = nahodneCislo(1, 2);
       for (int j = 0; j < pocetOddeleni; j++) {
         pozice = new Pozice(VEDOUCI_ODDELENI, zamestnanci.najdi(indexZamestnanec++));
         pobocka.vlozPozici(pozice);
         pobocka.zpristupniPodrizenouPozici(j + 1);
         int pocetPracovniku = nahodneCislo(1, 2);
         for (int k = 0; k < pocetPracovniku; k++) {
           pozice = new Pozice(PRACOVNIK, zamestnanci.najdi(indexZamestnanec++));
           pobocka.vlozPozici(pozice);
         }
         pobocka.zpristupniNadrizenouPozici();
       }
       pobocka.zpristupniNadrizenouPozici();
    }
    return pobocka;
  }
  public static IFirma generujFirmu(IZamestnanci zamestnanci)
        throws IllegalAccessException {
    IFirma firma = new Firma();
    IPobocka pobocka;
    for (int i = 0; i < 4; i++) {
      pobocka = generujPobocku(zamestnanci);
      firma.vloz(pobocka.getNazev(), pobocka);
    }
    return firma;
  }
  private static int nahodneCislo(int min, int max) {
    return (int) (Math.round(Math.random()) * (max - min) + min);
  }
}
```

Poznámka: Lorem ipsum (zkráceně lipsum) je označení pro standardní pseudolatinský text užívaný v grafickém designu a navrhování jako demonstrativní výplňový text při vytváření pracovních ukázek grafických návrhů (např. internetových stránek, rozvržení časopisů či všech druhů reklamních materiálů). Lipsum tak pracovně znázorňuje text v ukázkových maketách (tzv. mock-up) předtím, než bude do hotového návrhu vložen smysluplný obsah. (převzato z http://www.lorem-ipsum.cz/)

Ukázka použití generátoru dat

```
1    generujFirmu(generujZamestnance())
2      .stream()
3      .forEach(t -> {
4         System.out.println("\n"+t);
5         t.stream()
6         .forEach(p -> System.out.println(p));
7       }
8   );
```

Ukázka vygenerovaného obsahu firmy s pobočkami:

```
Pobocka{nazev=homero, mesto=Dry Prong, pocet pozic=11}
ředitel id=1: Lacy McKnight, gordon.bryant@example.com
vedoucí úseku id=6: Felecia O'Neil, allyson.farley@example.com
vedoucí oddělení id=9: Grant Schneider, wilford.wiley@example.com
pracovník id=11: Kristie Kennedy, rickie.hatfield@example.com
pracovník id=10: Angel Young, forest.hansen@example.com
vedoucí oddělení id=7: Bertha Delgado, lesley.reilly@example.com
pracovník id=8: Theron Christensen, kenya.rodriguez@example.com
vedoucí úseku id=2: Terrie Barker, santiago.mueller@example.com
vedoucí oddělení id=3: Sidney Kerr, jeffry.mosley@example.com
pracovník id=5: Heidi Weeks, antonia.briggs@example.com
pracovník id=4: Leopoldo Sosa, forrest.howe@example.com

Pobocka{nazev=errem, mesto=Oakfuskudshi, pocet pozic=14}
ředitel id=12: Rosie King, lydia.fuller@example.com
vedoucí úseku id=20: Ronny Gay, tabitha.de.la.cruz@example.com
vedoucí oddělení id=23: Hilda Nicholson, marlene.wood@example.com
pracovník id=25: Salvador Floyd, carmine.mcguire@example.com
pracovník id=24: Stella Haney, velma.roman@example.com
vedoucí oddělení id=21: Ben Gonzalez, kristy.maddox@example.com
pracovník id=22: Misty Morrow, joel.castro@example.com
vedoucí úseku id=13: Marshall Butler, freddie.sullivan@example.com
vedoucí oddělení id=17: Cornelia Duke, saundra.martinez@example.com
pracovník id=19: Eleanor McCall, chris.hester@example.com
pracovník id=18: Edith Sandoval, kristopher.lyons@example.com
vedoucí oddělení id=14: Cleveland Wheeler, morris.riddle@example.com
pracovník id=16: Elise Barr, allyson.morin@example.com
pracovník id=15: Angelo Howell, marietta.key@example.com

Pobocka{nazev=saperet, mesto=Stonebridge, pocet pozic=8}
ředitel id=26: Norris O'Neil, willie.christensen@example.com
vedoucí úseku id=27: Jermaine Cochran, elliott.rosales@example.com
vedoucí oddělení id=31: Kimberley Carpenter, carmen.adkins@example.com
pracovník id=33: Tia Black, marc.wyatt@example.com
pracovník id=32: Isidro Burton, letitia.crawford@example.com
vedoucí oddělení id=28: Tessa Kemp, erin.sosa@example.com
pracovník id=30: Lynette Fuller, ophelia.herrera@example.com
pracovník id=29: Terra Workman, lily.brooks@example.com

Pobocka{nazev=dolores, mesto=Whitebreast, pocet pozic=12}
ředitel id=34: Etta Carver, jolene.grant@example.com
vedoucí úseku id=40: Bryce Rivas, danielle.wolf@example.com
```

vedoucí oddělení id=44: Dina Hebert, stacy.key@example.com
pracovník id=45: Terence Roberts, roxie.cervantes@example.com
vedoucí oddělení id=41: Lazaro Baxter, kelley.callahan@example.com
pracovník id=43: Andre Shaffer, elvira.kaufman@example.com
pracovník id=42: Eddy Cruz, maritza.vaughan@example.com
vedoucí úseku id=35: Lorraine Woods, hester.mathews@example.com
vedoucí oddělení id=38: Mason Stein, delbert.mayo@example.com
pracovník id=39: Juanita Winters, matthew.alvarez@example.com
vedoucí oddělení id=36: Fernando Evans, silas.foley@example.com
pracovník id=37: Brain Cameron, lauri.gallagher@example.com