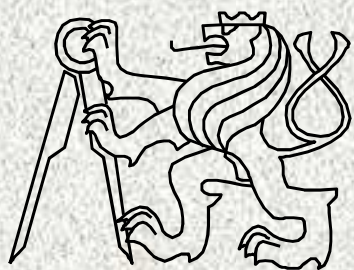


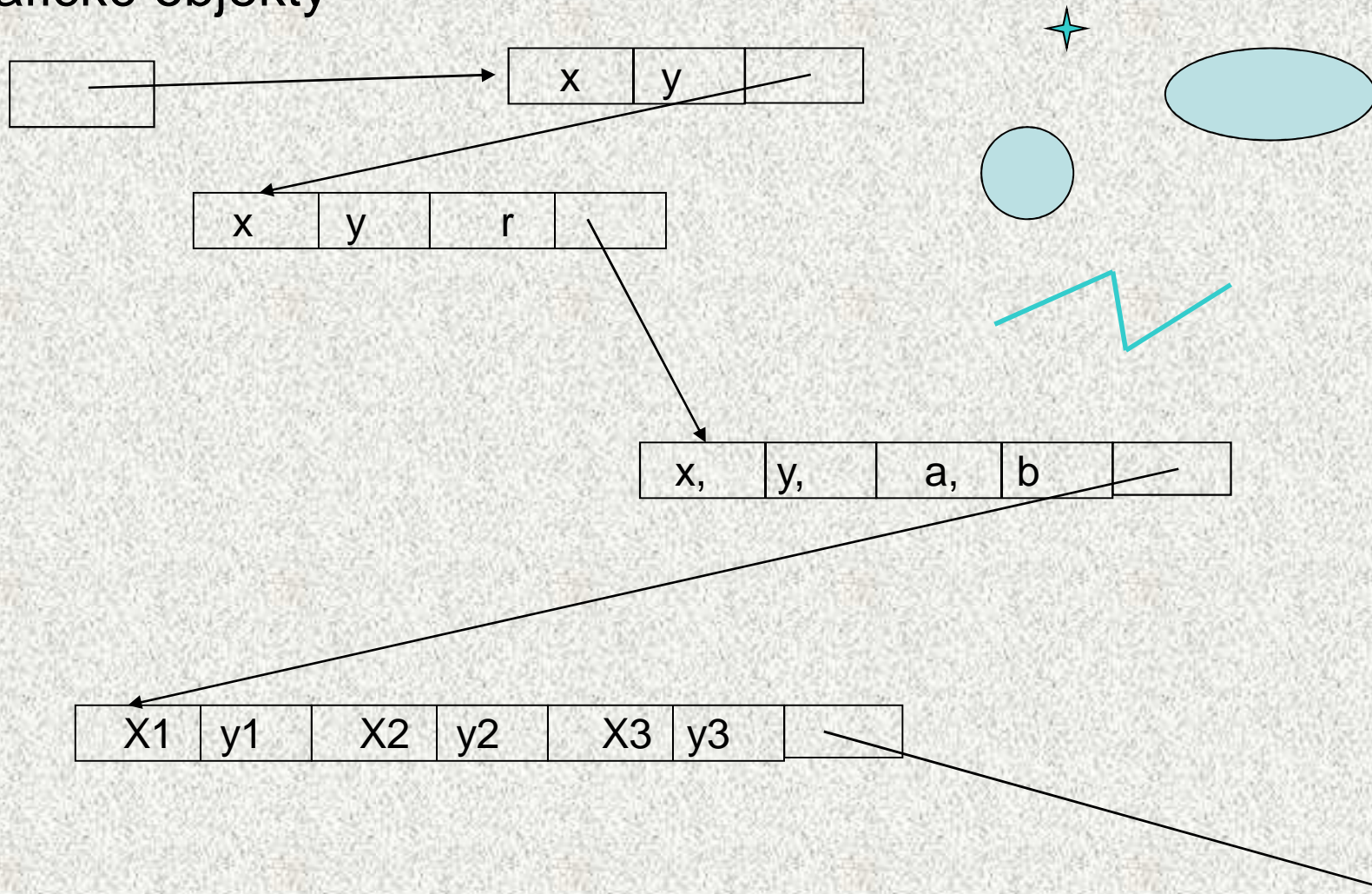
Spojové struktury



A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické

Spojové struktury

- Grafické objekty

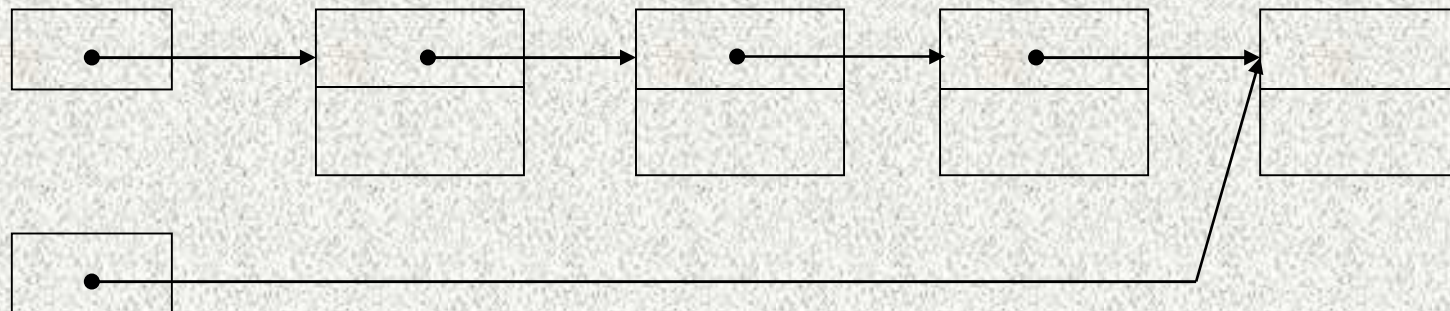


Spojové struktury

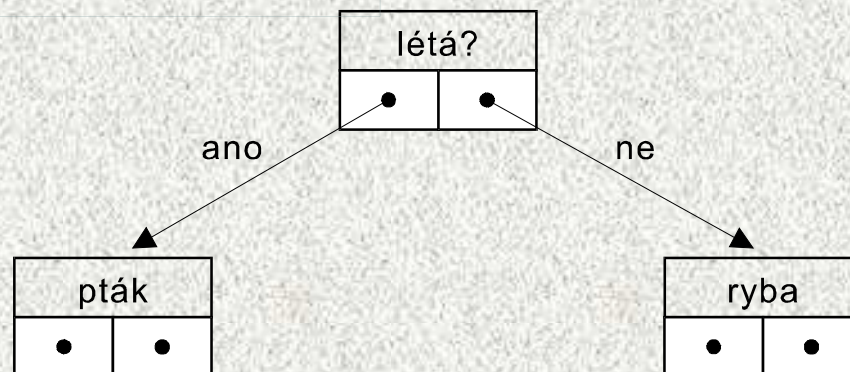
Zásobník



Fronta

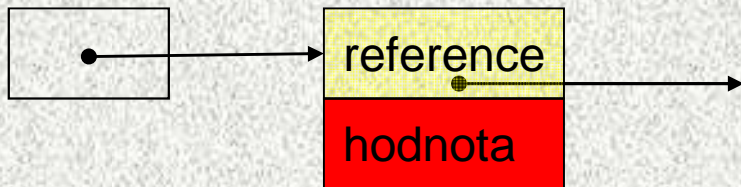


Stromy



Spojové struktury

- Základním prvkem spojových struktur je dvojice:



- **reference** - odkaz(y) na další prvek spojové struktury, definuje operátor **new**
 - znázorňujeme šipkou
- **hodnota** – informace libovolného typu
- Nejjednodušší typ spojové struktury – **jednosměrné spojové seznamy**
- Spojové struktury jsou mocným implementačním prostředkem pro ADT, viz další přednáška o ADT a předmět Algoritmizace:
 - Fronty
 - Zásobníky
 - Stromy
 - Grafy

Spojové seznamy I

Prvek seznamu:

```
class Prvek {  
    TypHodnoty hodn;  
    Prvek dalsi;  
}
```

Rekurzivní definice

hodn

dalsi



```
public Prvek(TypHodnoty h, Prvek p) {  
    hodn = h;  
    dalsi = p;  
}
```

```
Prvek hlava = null;
```

```
hlava = new Prvek(14, null);
```

```
Prvek prvek1;
```

```
prvek1 = new Prvek(22, null);
```

hlava



hlava



prvek1

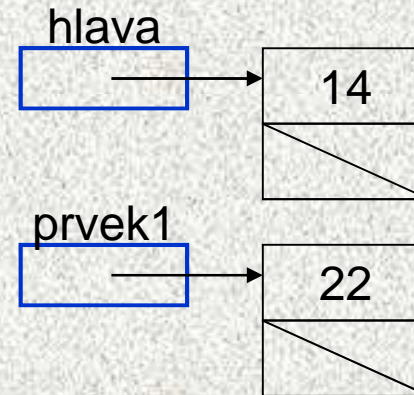


Spojové seznamy II

```
hlava = new Prvek(14, null);
```

```
prvek1 = new Prvek(22, null);
```

```
prvek1 = hlava;
```

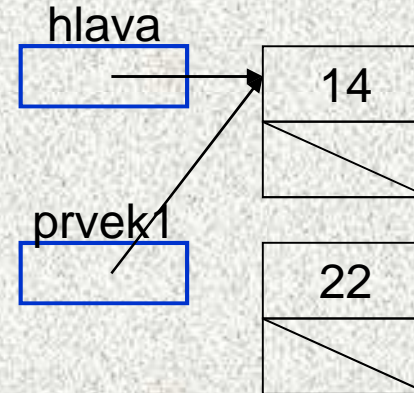


Spojové seznamy II

```
hlava = new Prvek(14, null);
```

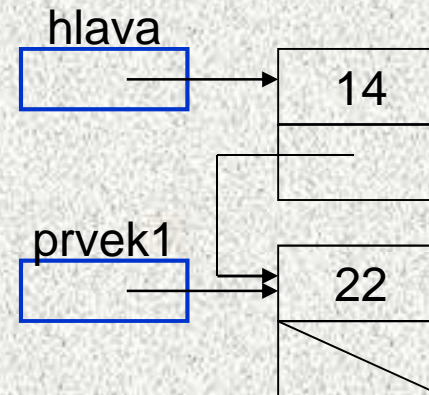
```
prvek1 = new Prvek(22, null);
```

```
prvek1 = hlava;
```



```
prvek1 = new Prvek(22, null);
```

```
hlava = new Prvek(14, prvek1);
```

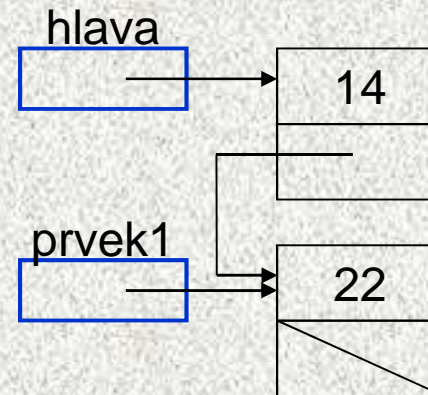


Spojové seznamy III

```
prvek1 = new Prvek(22, null);
```

```
hlava = new Prvek(14, prvek1);
```

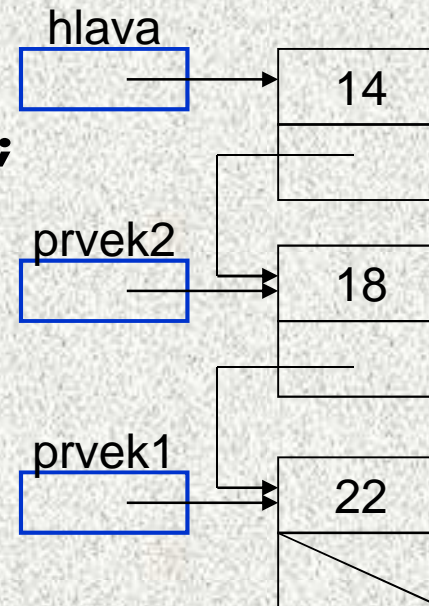
```
Prvek prvek2;
```



```
prvek1 = new Prvek(22, null);
```

```
prvek2 = new Prvek(18, prvek1);
```

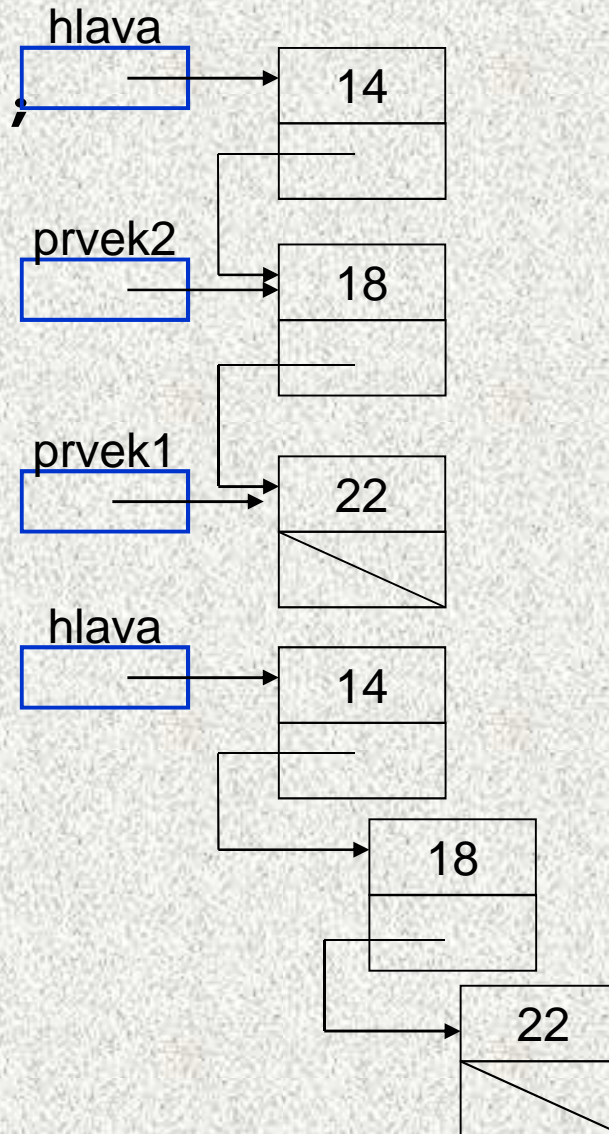
```
hlava = new Prvek(14, prvek2);
```



Spojové seznamy IV

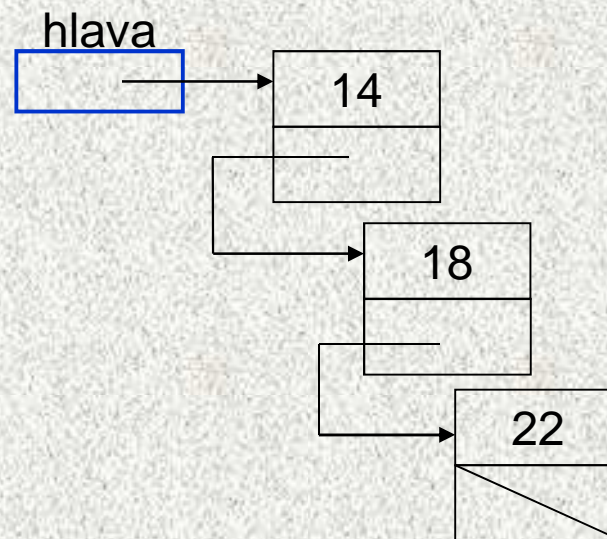
```
prvek1 = new Prvek (22,null);  
prvek2 = new Prvek (18,prvek1);  
hlava = new Prvek (14,prvek2);
```

```
hlava = new Prvek (14,  
                  new Prvek (18,  
                              new Prvek (22, null)  
                              )  
                  );
```

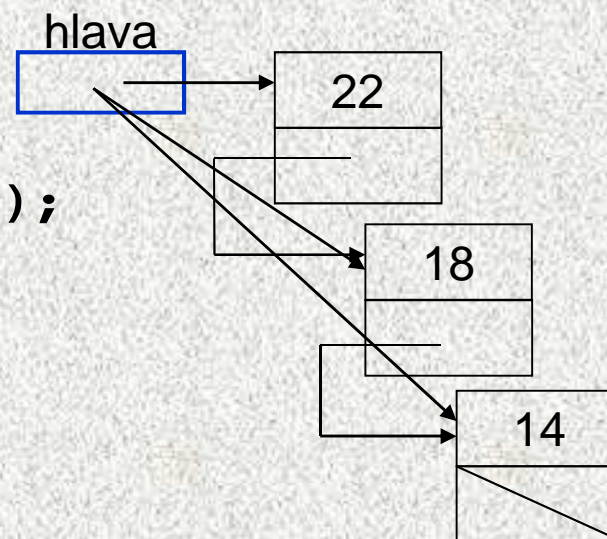


Spojové seznamy V

```
hlava = new Prvek(14,  
    new Prvek(18,  
        new Prvek(22, null)  
    )  
);
```



```
Prvek hlava = null;  
int cislo= 14;  
while (cislo<26) {  
    hlava = new Prvek(cislo,hlava);  
    cislo += 4;  
}
```



Spojové seznamy VI

```
public int hodn() {  
    return hodn;  
}
```

```
Prvek hlava;
```

```
System.out.print(hlava.hodn());
```

```
System.out.println(hlava.dalsi.hodn());
```

```
System.out.println(hlava.dalsi.dalsi.hodn());
```

```
public Prvek dalsi() {  
    return dalsi; }  
Prvek pom = hlava;
```

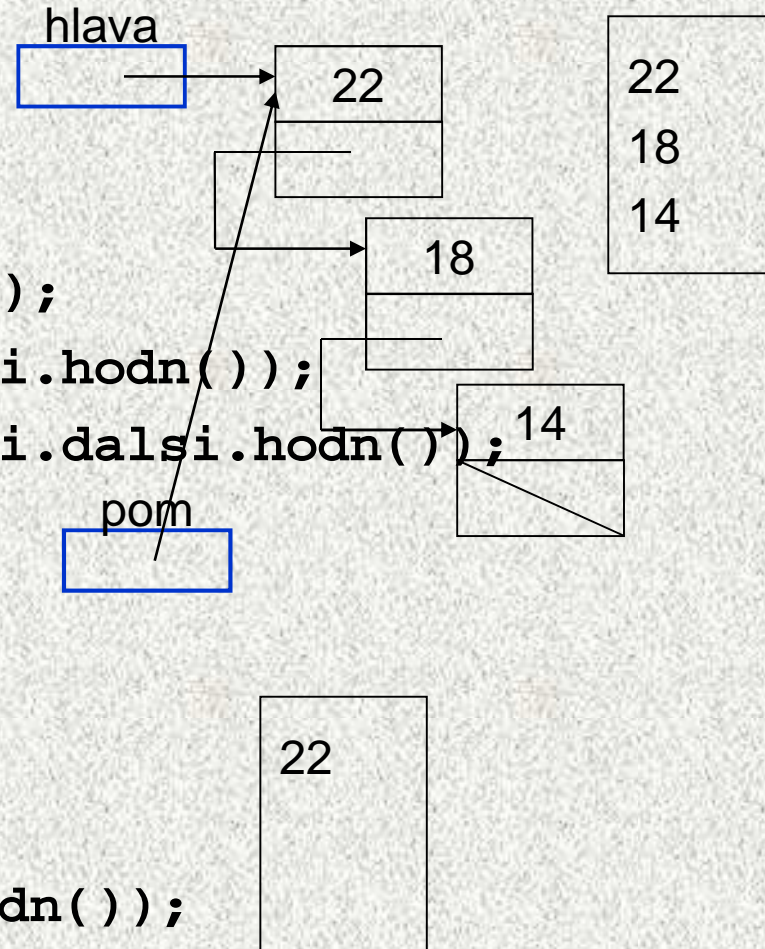
```
while (pom!=null) {
```

```
    System.out.println(pom.hodn());
```

```
    pom = pom.dalsi();
```

```
}
```

A0B36PR1 - 10



Spojové seznamy VI

```
public int hodn() {  
    return hodn;  
}
```

```
Prvek hlava;
```

```
System.out.print(hlava.hodn());
```

```
System.out.println(hlava.dalsi.hodn());
```

```
System.out.println(hlava.dalsi.dalsi.hodn());
```

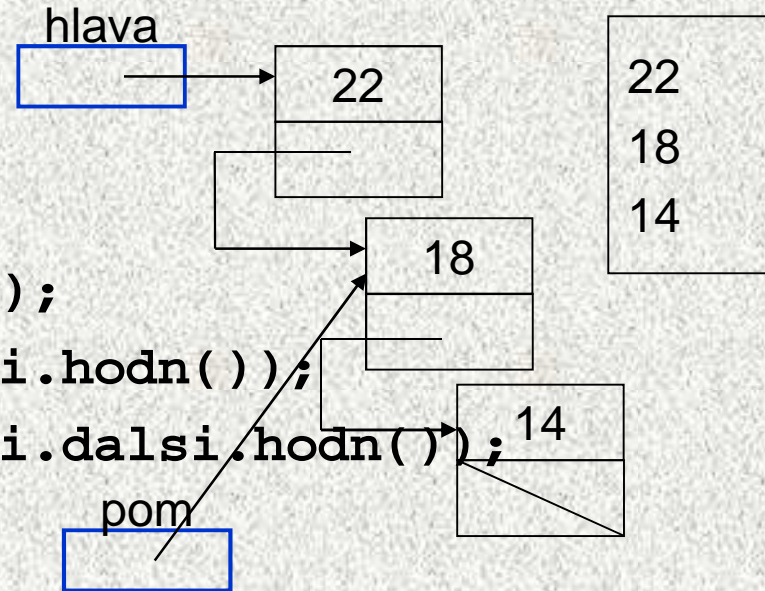
```
public Prvek dalsi() {  
    return dalsi; }  
Prvek pom = hlava;
```

```
while (pom!=null) {
```

```
    System.out.println(pom.hodn());
```

```
    pom = pom.dalsi();
```

```
}
```



Spojové seznamy VI

```
public int hodn() {  
    return hodn;  
}
```

```
Prvek hlava;
```

```
System.out.print(hlava.hodn());
```

```
System.out.println(hlava.dalsi.hodn());
```

```
System.out.println(hlava.dalsi.dalsi.hodn());
```

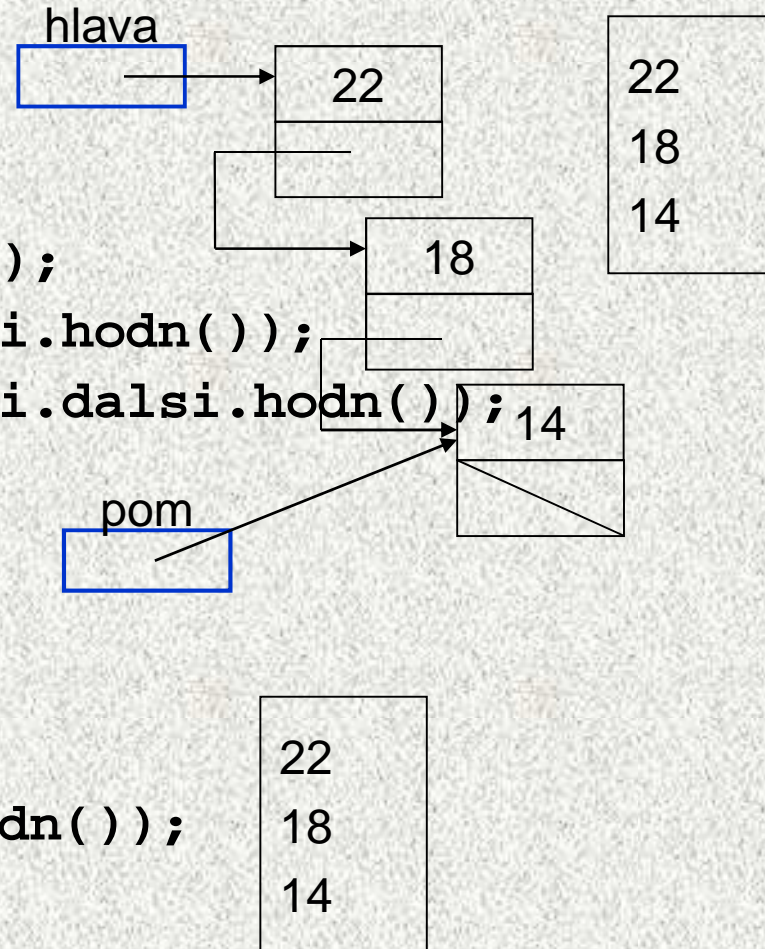
```
public Prvek dalsi() {  
    return dalsi; }  
Prvek pom = hlava;
```

```
while (pom!=null) {
```

```
    System.out.println(pom.hodn());
```

```
    pom = pom.dalsi();
```

```
}
```

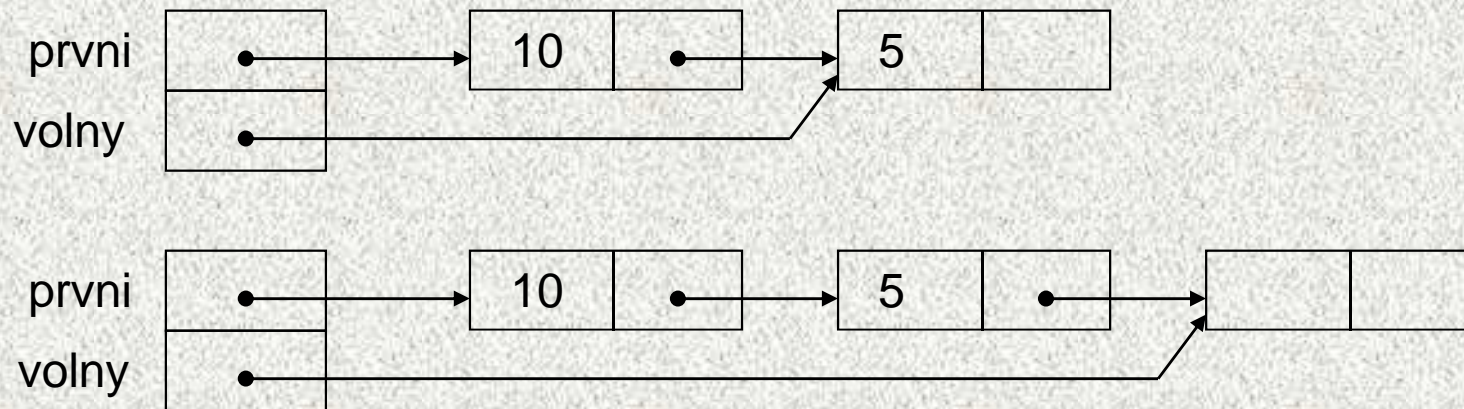


Příklad použití spojového seznamu

```
public class ObraceniCisel {
    public static void main(String[] args) {
        System.out.println("zadejte řadu zakončených nulou");
        Prvek prvni = null;
        int cislo = sc.nextInt();
        while (cislo!=0) {
            prvni = new Prvek(cislo, prvni);
            cislo = sc.nextInt();
        }
        System.out.println("čísla v opačném pořadí");
        Prvek pom = prvni;
        while (pom!=null) {
            System.out.print(pom.hodn()+" ");
            pom = pom.dalsi();
        }
        System.out.println();
    }
}
```

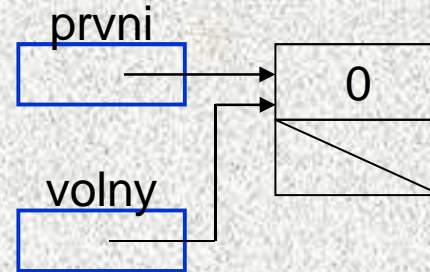
Další operace se spojovým seznamem

- Napišme třídu reprezentující spojový seznam celých čísel s těmito operacemi:
 - vložení čísla na začátek seznamu
 - vložení čísla na konec seznamu
 - test, zda číslo je v seznamu
 - výpis čísel uložených v seznamu
- Vložení prvku na konec spojového seznamu identifikovaného pouze odkazem na první prvek vyžaduje najít poslední prvek (lineární složitost)
- Vložení prvku na konec seznamu bude mít konstantní složitost, použijeme-li reprezentaci:



Spojové seznamy VII

```
class SeznamCisel {  
    Prvek volny;  
    Prvek prvni;  
  
    public SeznamCisel() {  
        prvni = new Prvek();  
        volny = prvni;  
    }  
}
```

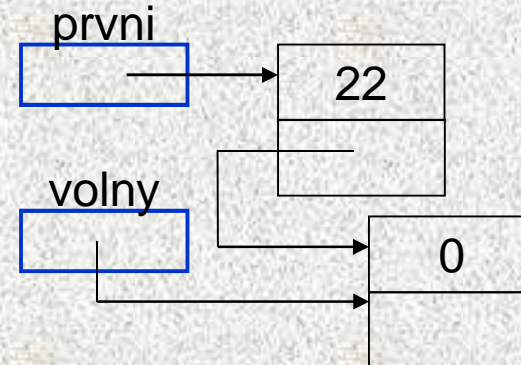


```
public Prvek() {  
    hodn = 0;  
    dalsi = null;  
}
```

```
SeznamCisel jednodusek = new SeznamCisel();
```

Spojové seznamy VII

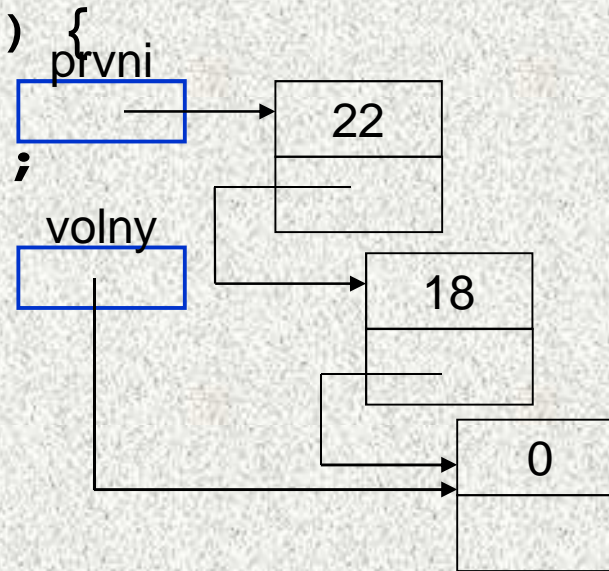
```
public void vlozNaKonec(int x) {  
    volny.hodn = x;  
    volny.dalsi = new Prvek();  
    volny = volny.dalsi;  
}
```



```
SeznamCisel jednoduseK = new SeznamCisel();  
jednoduseK.vlozNaKonec(22);
```


Spojové seznamy VII

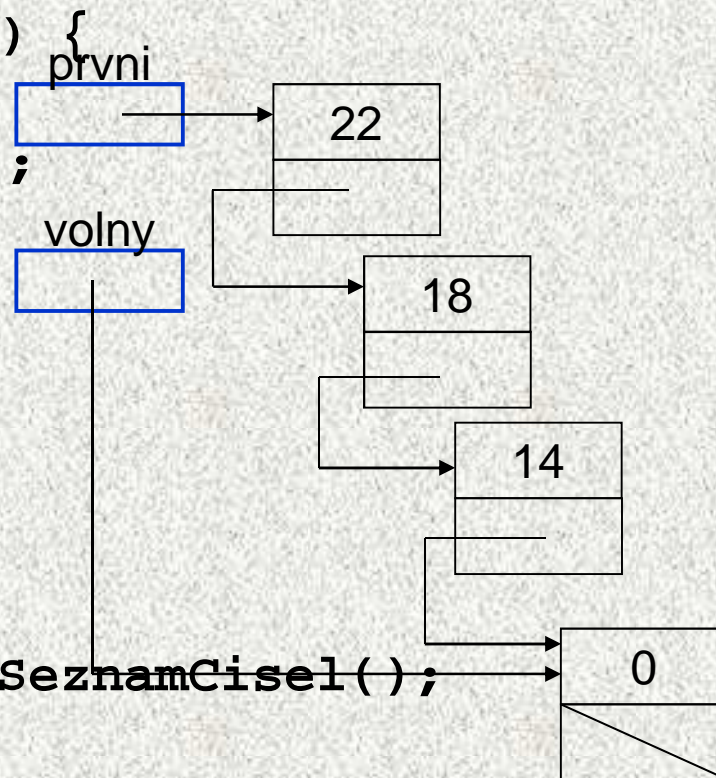
```
public void vlozNaKonec(int x) {  
    volny.hodn = x;  
    volny.dalsi = new Prvek();  
    volny = volny.dalsi;  
}
```



```
SeznamCisel jednoduseK = new SeznamCisel();  
jednoduseK.vlozNaKonec(22);  
jednoduseK.vlozNaKonec(18);
```


Spojové seznamy VII

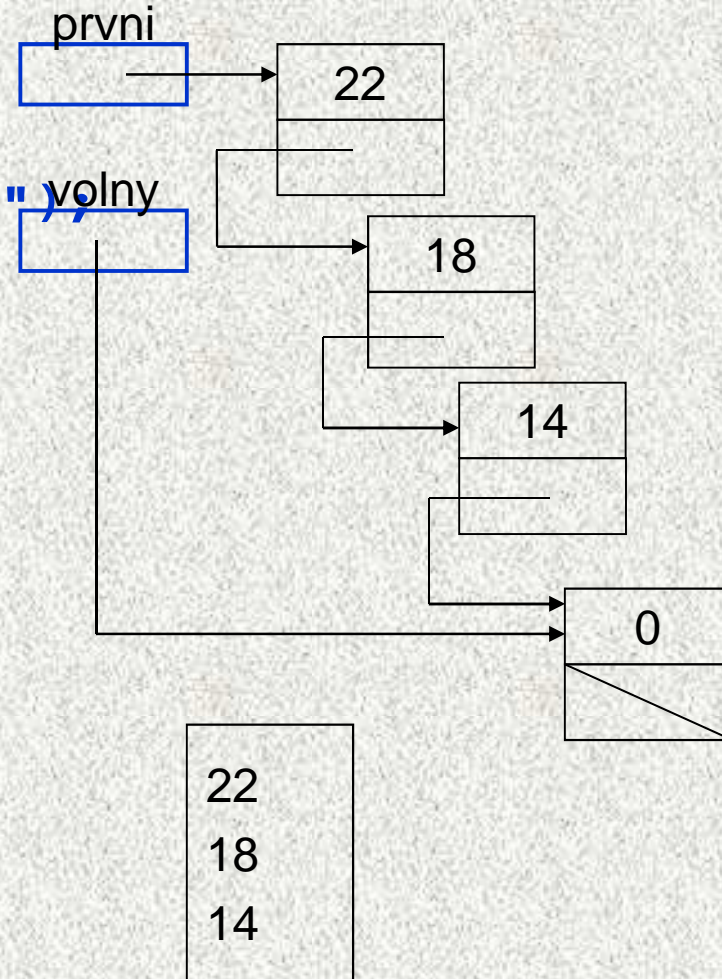
```
public void vlozNaKonec(int x) {  
    volny.hodn = x;  
    volny.dalsi = new Prvek();  
    volny = volny.dalsi;  
}
```



```
SeznamCisel jednoduseK = new SeznamCisel();  
jednoduseK.vlozNaKonec(22);  
jednoduseK.vlozNaKonec(18);  
jednoduseK.vlozNaKonec(14);
```

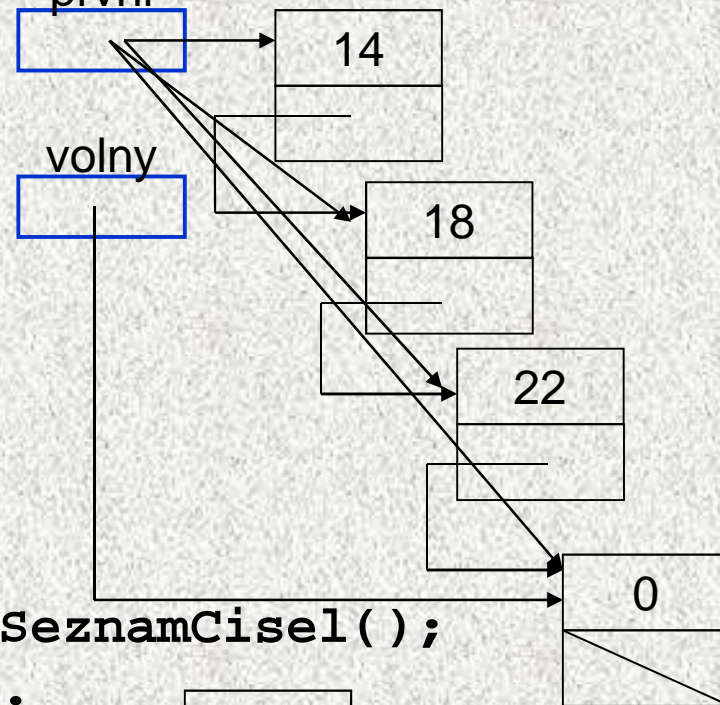
Spojové seznamy VII

```
public void vypis() {  
    Prvek pom = prvni;  
    while (pom!=volny) {  
        System.out.print(pom.hodn+" ");  
        pom = pom.dalsi;}  
    System.out.println();  
}
```



Spojové seznamy VIII

```
public void vlozNaZacatek(int x) {  
    prvni = new Prvek(x, prvni);  
}
```



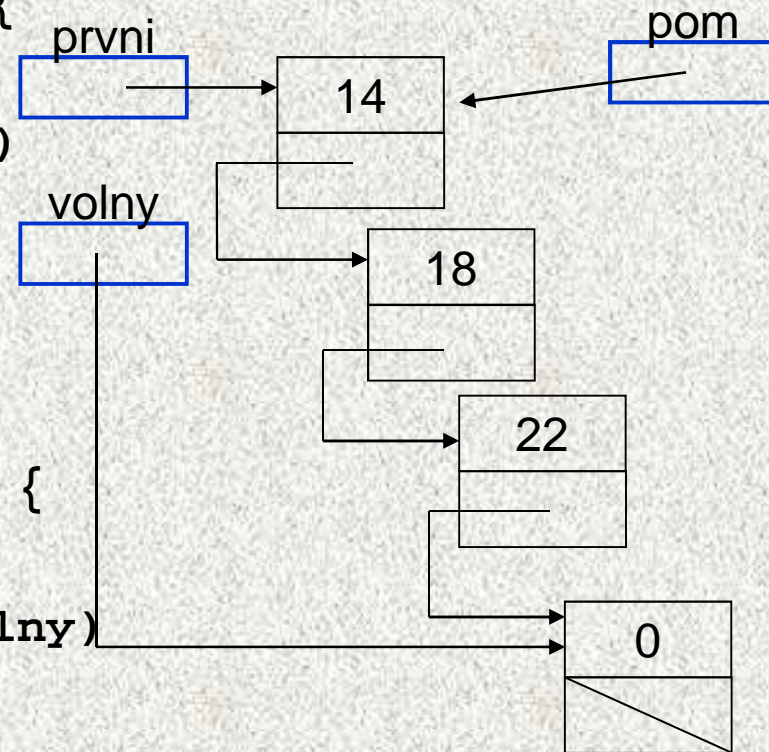
```
SeznamCisel jednoduseK = new SeznamCisel();  
jednoduseZ.vlozNaZacatek(22);  
jednoduseZ.vlozNaZacatek(18);  
jednoduseZ.vlozNaZacatek(14);  
jednoduseZ.vypis();
```

```
14  
18  
22
```

Spojové seznamy IX

```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn!=x && pom!=volny)  
        pom = pom.dalsi;  
    return pom!=volny;  
}
```

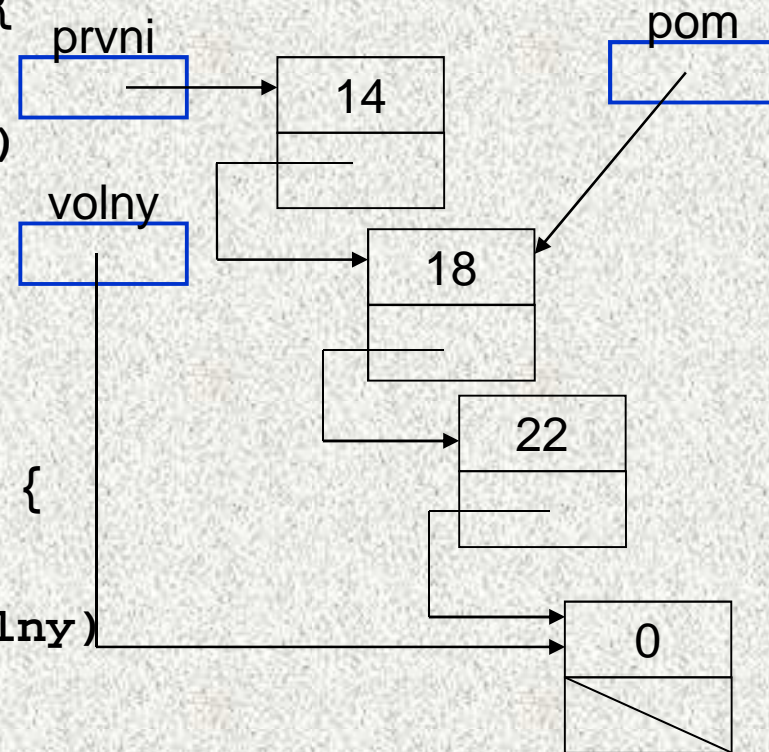
```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi();  
    return pom!=volny;  
}
```



Spojové seznamy IX

```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn!=x && pom!=volny)  
        pom = pom.dalsi;  
    return pom!=volny;  
}
```

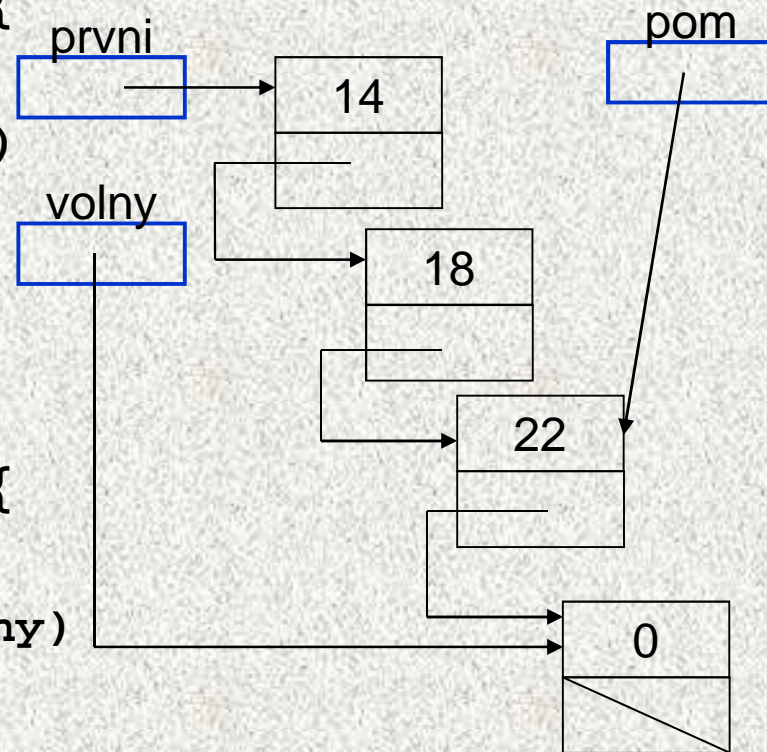
```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi();  
    return pom!=volny;  
}
```



Spojové seznamy IX

```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi;  
    return pom!=volny;  
}
```

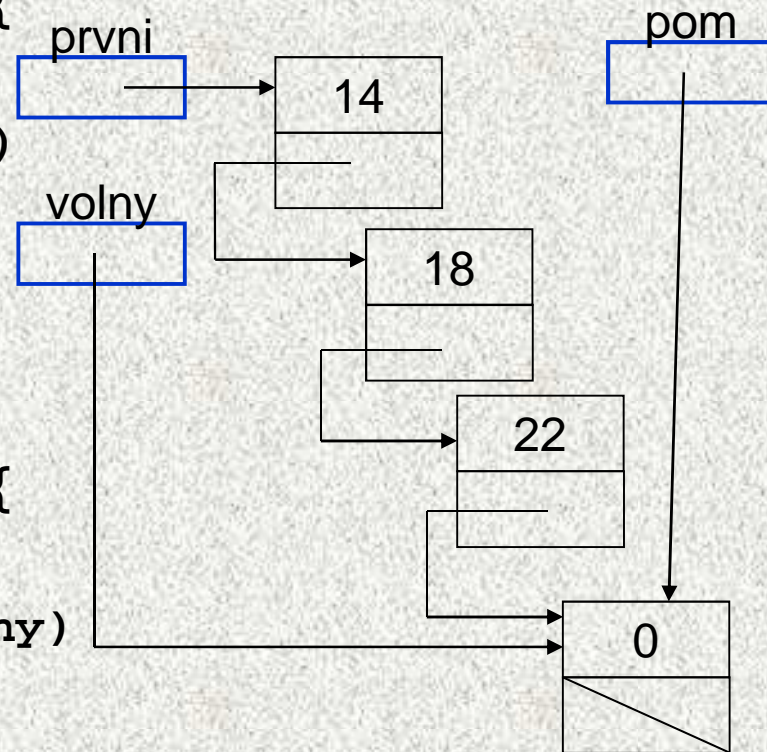
```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi();  
    return pom!=volny;  
}
```



Spojové seznamy IX

```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi;  
    return pom!=volny;  
}
```

```
public boolean jePrvkem1(int x) {  
    Prvek pom = prvni;  
    while (pom.hodn()!=x && pom!=volny)  
        pom = pom.dalsi();  
    return pom!=volny;  
}
```



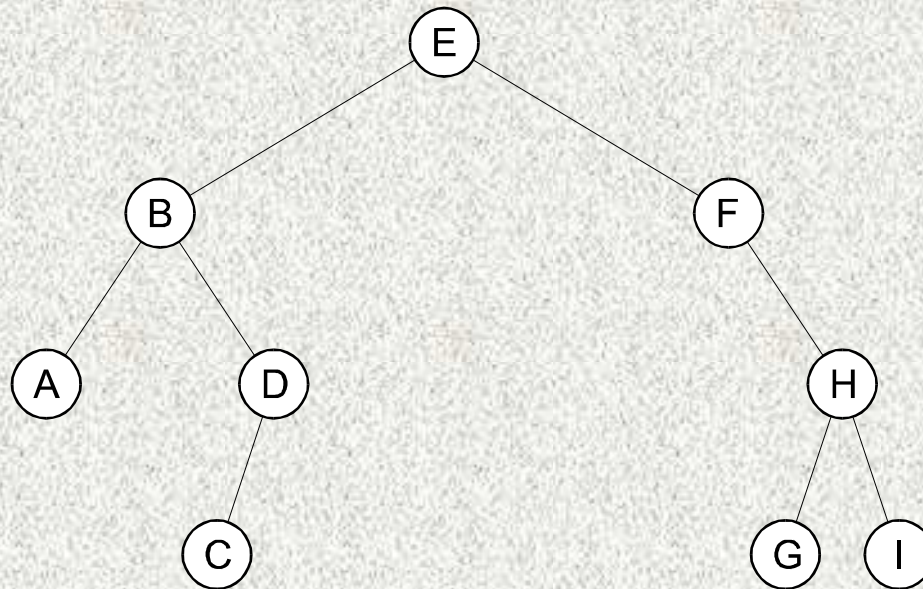
Použití třídy SeznamCisel

- Program, který přečte řadu čísel zakončených nulou a vypíše:
 - přečtená čísla v opačném pořadí
 - seznam různých čísel

```
public static void main(String[] args) {
    SeznamCisel obracena = new SeznamCisel();
    SeznamCisel ruzna = new SeznamCisel();
    System.out.println ("zadejte řadu zakončenou nulou");
    int x = sc.nextInt();
    while (x!=0) {
        obracena.vlozNaZacatek(x);
        if (!ruzna.jePrvkem(x))
            ruzna.vlozNaKonec(x);
        x = sc.nextInt();
    }
    System.out.println ("čísla v opačném pořadí");
    obracena.vypis();
    System.out.println ("seznam různých čísel");
    ruzna.vypis();
}
```

Stromy

- Lineární spojová struktura (spojový seznam)
 - každý prvek má nanejvýš jednoho následníka
- Nelineární spojová struktura (strom):
 - každý prvek může mít více následníků
- Binární strom: každý prvek (uzel) má nanejvýš dva následníky



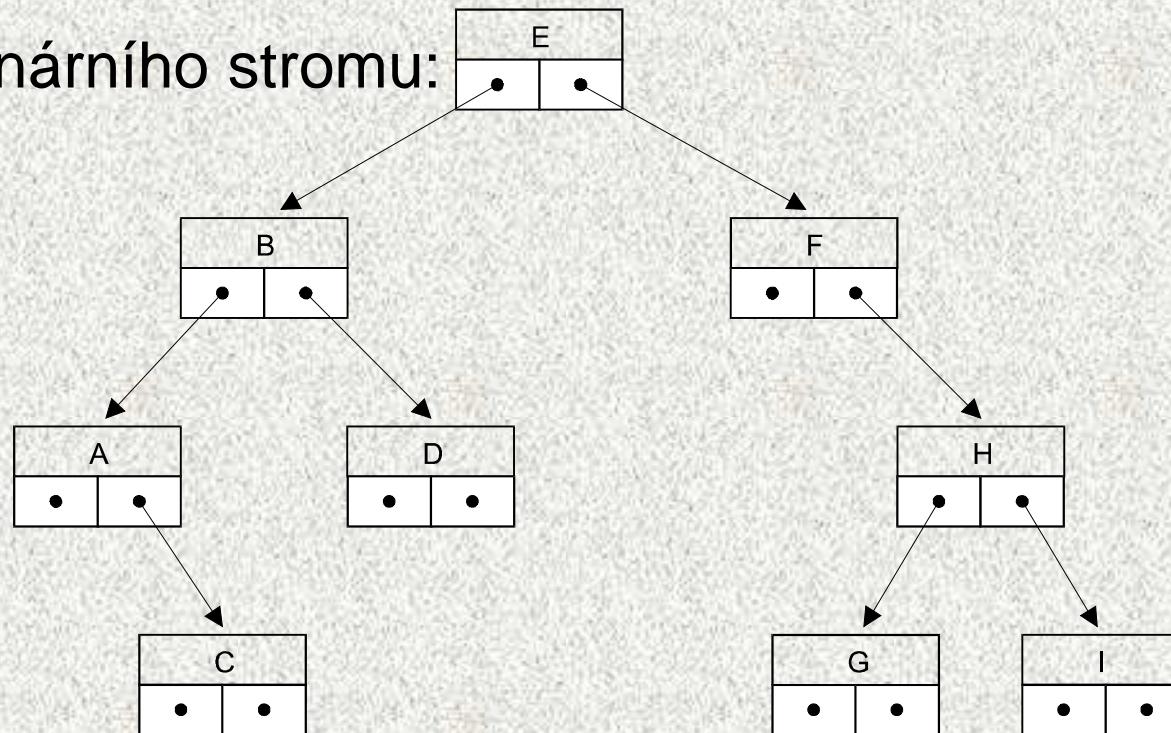
- Některé pojmy: kořen stromu, levý podstrom, pravý podstrom, list

Realizace binárního stromu

- Třída pro realizaci:

```
class Uzel {  
    char hodn;  
    Uzel levy, pravy;  
};
```

- Příklad binárního stromu:



Příklad - hra „Jaké zvíře si myslíš“

- Příklad dialogu:

Myslíte si nějaké zvíře?

Ano

Zvíře, které si myslíte létá?

Ne

Je to ryba?

Ne

Dám se podat. Jaké zvíře jste myslel?

Pes

Napište otázku vystihující rozdíl mezi pes a ryba!

štěká?

Pro zvíře, které jste myslel, je odpověď ano či ne?

ano

Dekuji.

Chcete hrát ještě jednou?

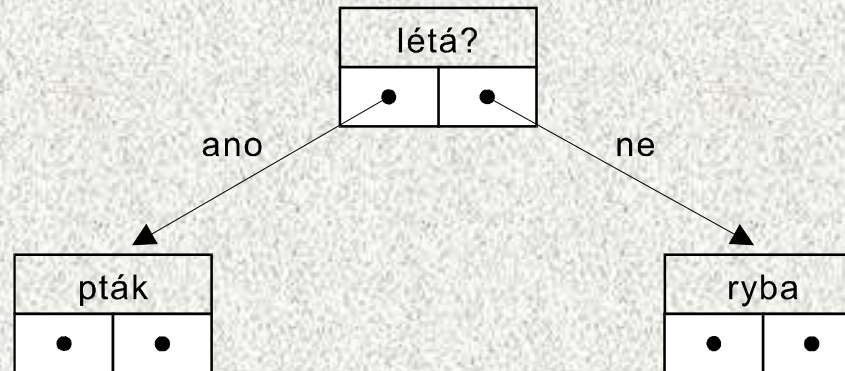
Ano

Myslíte si nějaké zvíře?

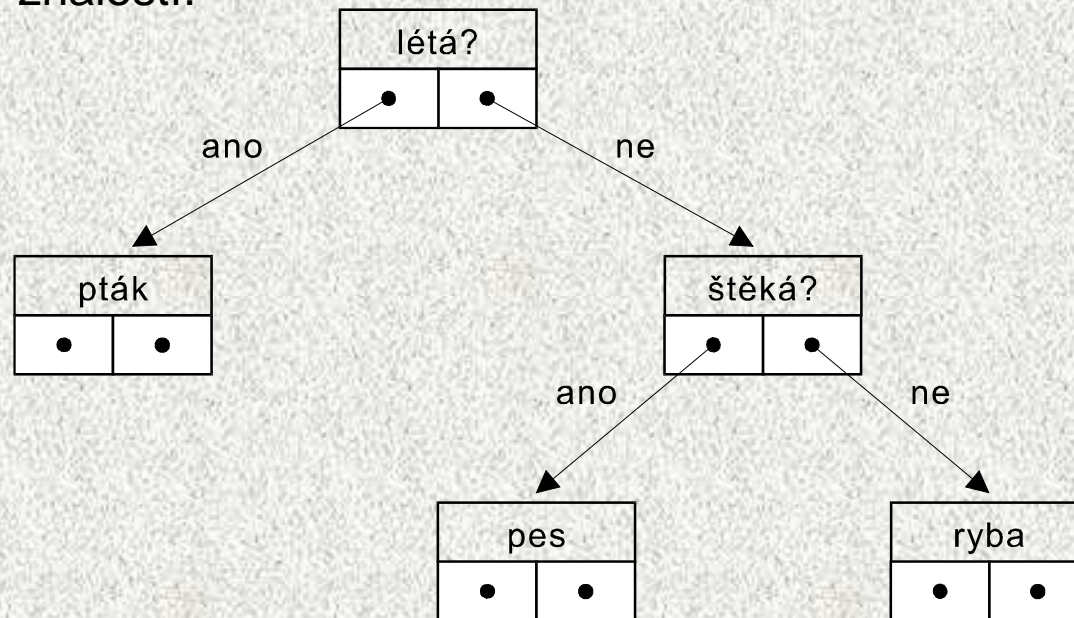
Ano

Příklad - hra „Jaké zvíře si myslíš“

- Počáteční strom dialogu :



- Strom dialogu po doplnění znalostí:



Příklad - hra „Jaké zvíře si myslíš“

- Hrubé řešení:

```
"úvod dialogu";
```

```
"aktuálním uzlem je kořen stromu";
```

```
do {
```

```
    "polož otázku uvedenou v aktuálním uzlu";
```

```
    if ("odpověď je ano")
```

```
        "aktuálním uzlem je levý následník"
```

```
    else
```

```
        "aktuálním uzlem je pravý následník"
```

```
    } while ("aktuální uzel není list");
```

```
"polož závěrečnou otázku, název zvířete vyber z  
    aktuálního uzlu";
```

```
if ("odpověď je ano")
```

```
    "hádání bylo úspěšné"
```

```
else {
```

```
    "hádání bylo neúspěšné"; "doplň znalosti"}
```


Příklad - hra „Jaké zvíře si myslíš“

- Podrobné řešení

```
class Uzel {
    String text;
    Uzel ano, ne;;
    public Uzel(String t) {
        text = t; ano = null; ne = null;
    }
    public Uzel(String t, Uzel a, Uzel n) {
        text = t; ano = a; ne = n;
    }
    public boolean jeList() {
        return ano==null && ne==null;
    }
}
```

Příklad - hra „Jaké zvíře si myslíš“

- Hlavní funkce:

```
public class Hra {
    public static void main(String[] args) {
        Uzel koren = inicializaceStromu();
        for (;;) {
            System.out.println ("Myslíte si nějaké zvíře?");
            if (!odpovedAno()) break;
            Uzel aktualni = koren;
            do {System.out.println(aktualni.text);
                if (odpovedAno()) aktualni = aktualni.ano;
                else aktualni = aktualni.ne;
            } while (!aktualni.jeList());
            System.out.println("Je to "+aktualni.text+"?");
            if (odpovedAno()) System.out.println("Uhádl jsem");
            else {
                System.out.println("Neuhádl jsem. Prosím o doplnění");
                doplnPodstrom(aktualni);
            }
            System.out.println("Děkuji. Chcete pokračovat?");
            if (!odpovedAno()) break;
        }
    }
}
```

Příklad - hra „Jaké zvíře si myslíš“

- Pomocné funkce:

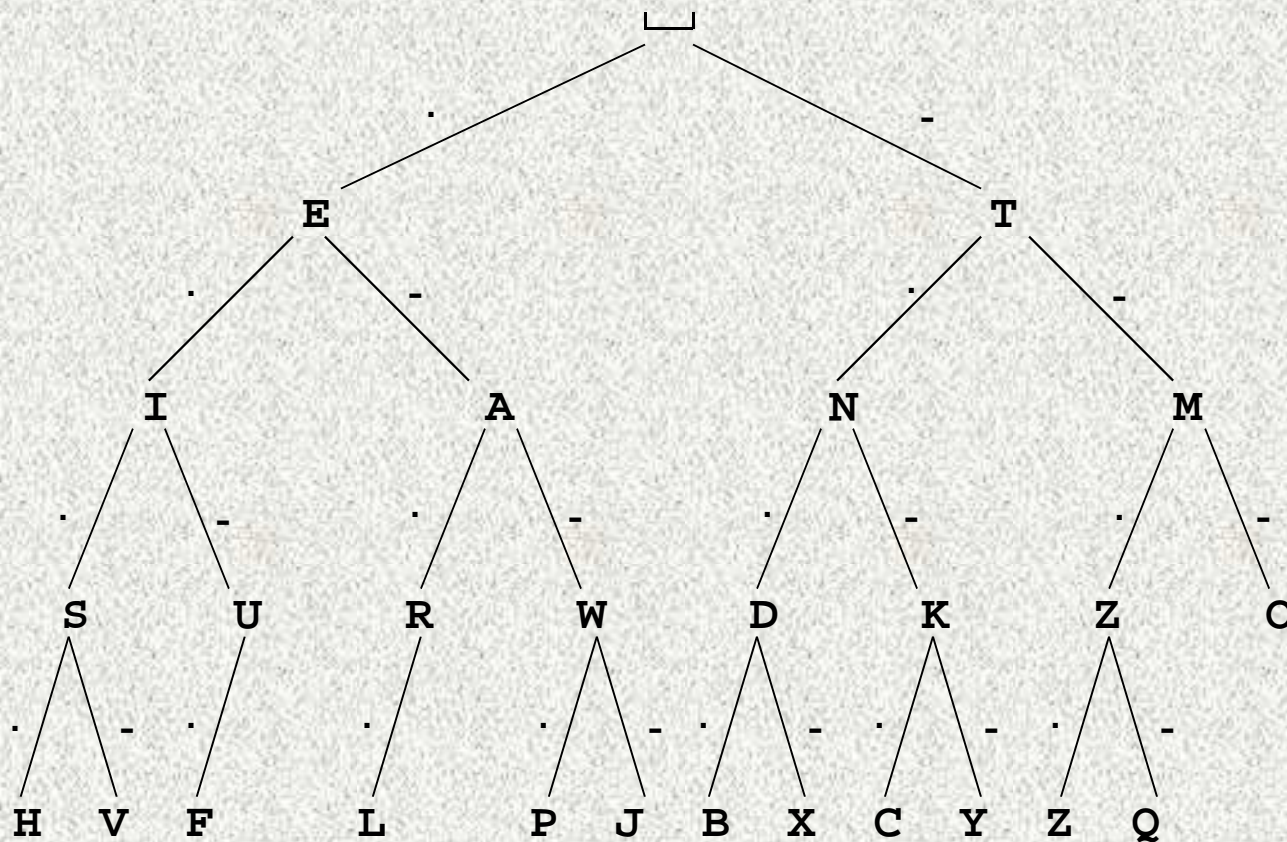
```
static boolean odpovedAno() {
    String text = sc.nextLine();
    if (s.length()>0 &&
        (s.charAt(0)=='a' || s.charAt(0)=='A'))
        return true;
    else
        return false;
}
static Uzel inicializaceStromu() {
    return new Uzel("létá?",
                    new Uzel("pták", null, null),
                    new Uzel("ryba", null, null));
}
```

Příklad - hra „Jaké zvíře si myslíš“

```
static void doplnPodstrom(Uzel p) {
Scanner sc = new Scanner(System.in);
String noveZvire, novaOtazka;
Uzel novyAno, novyNe;
System.out.println("Jaké zvíře jste myslel?");
noveZvire = sc.nextLine();
System.out.println(" ... rozdíl mezi " + noveZvire + " a " + p.text);
novaOtazka = sc.nextLine();
System.out.println("Pro zvíře(" + noveZvire + "), odpověď ano či ne");
if (odpovedAno()) {
    novyAno = new Uzel(noveZvire);
    novyNe = new Uzel(p.text);
} else {
    novyAno = new Uzel(p.text);
    novyNe = new Uzel(noveZvire);
}
p.text = novaOtazka;
p.ano = novyAno;
p.ne = novyNe;
}
```


Příklad – dekódování morseovky

- Pro dekódování textu zapsaného v Morseově abecedě lze použít následující binární strom



Příklad – dekódování morseovky

- Strom vytvoříme z objektů typu *MUzel*

```
class MUzel {  
    char znak;  
    MUzel tecka, carka;  
  
    public MUzel(char z) {  
        znak = z; tecka = null; carka = null;  
    }  
  
    public MUzel(char z, MUzel t, MUzel c) {  
        znak = z; tecka = t; carka = c;  
    }  
}
```

Příklad – dekódování morseovky

- Pro vytvoření stromu zavedeme funkci:

```
static MUzel strom() {  
    return  
        new MUzel(' ',  
            new MUzel('E', // .  
                new MUzel('I', // ..  
                    new MUzel('S', // ...  
                        new MUzel('H'), // ....  
                        new MUzel('V') // ...-  
                    ),  
                new MUzel('U', // ..-  
                    new MUzel('F'), // ..-.  
                    null // ..-  
                ),  
            ),  
        new MUzel('A', // .-  
            ...  
        ),  
        new MUzel('T', // -  
            ...  
        )  
    }  
}
```

Příklad – dekódování morseovky

```
public static void main(String[] args) {  
    System.out.println ("Zadejte text v morseovce  
                        zakončený prázdným řádkem");  
    String text = sc.nextLine();  
    while (!text.equals("")) {  
        System.out.print(dekoduuj(text+" "));  
        text = sc.nextLine();  
    }  
}  
  
static MUzel koren = strom();
```


Příklad – dekódování morseovky

- Dekódování zapíšeme jako funkci, jejímž parametrem je řetězec obsahující Morseův kód a výsledkem je řetězec tvořený odpovídajícími znaky latinské abecedy

```
static String dekoduj(String s) {  
    MUzel aktualni = koren;  
    String vysl = "";  
    for (int i=0; i<s.length(); i++) {  
        char z = s.charAt(i);  
        if (aktualni!=null)  
            if (z=='.') aktualni = aktualni.tecka;  
            else if (z=='-') aktualni = aktualni.carka;  
            else {vysl = vysl + aktualni.znak;  
                aktualni = koren;}  
        else {vysl = vysl + '?';  
            aktualni = koren;}  
        }  
        return vysl;  
    }  
}
```