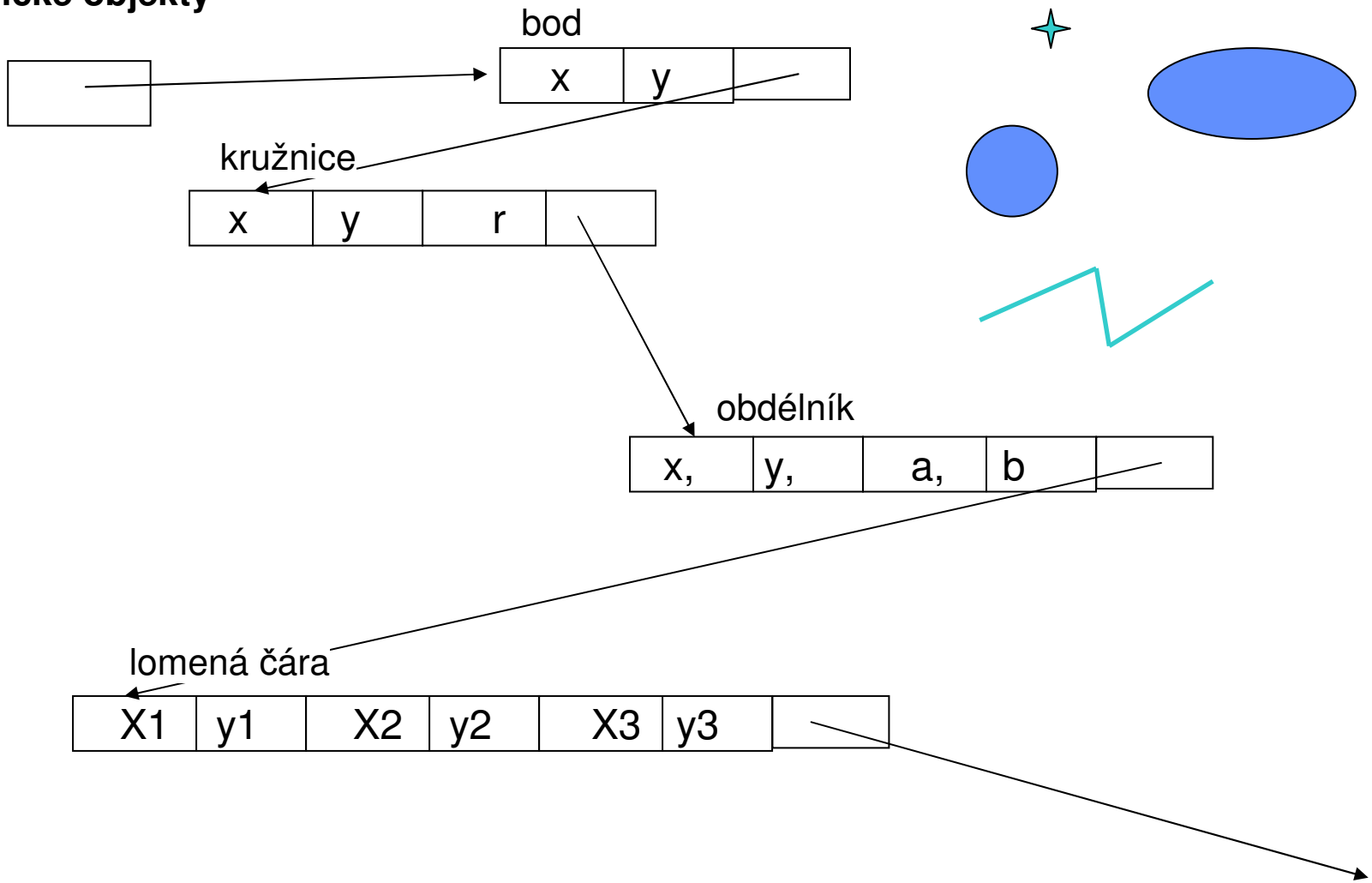
Spojové struktury

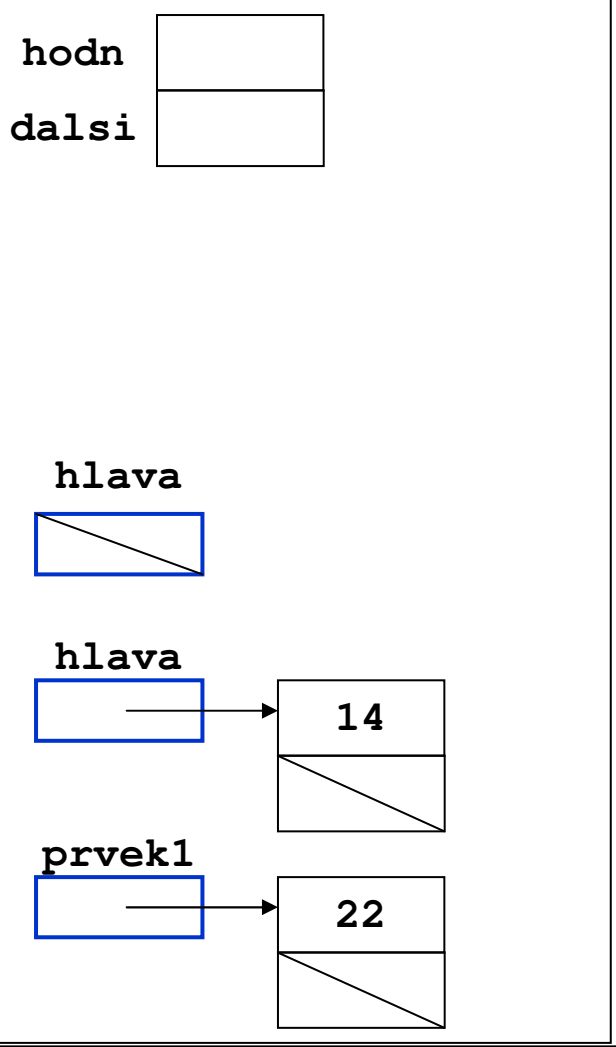
- **Grafické objekty**



Spojové seznamy I

Prvek seznamu:

```
class Prvek {
    TypHodnoty hodn;
    Prvek 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);
```



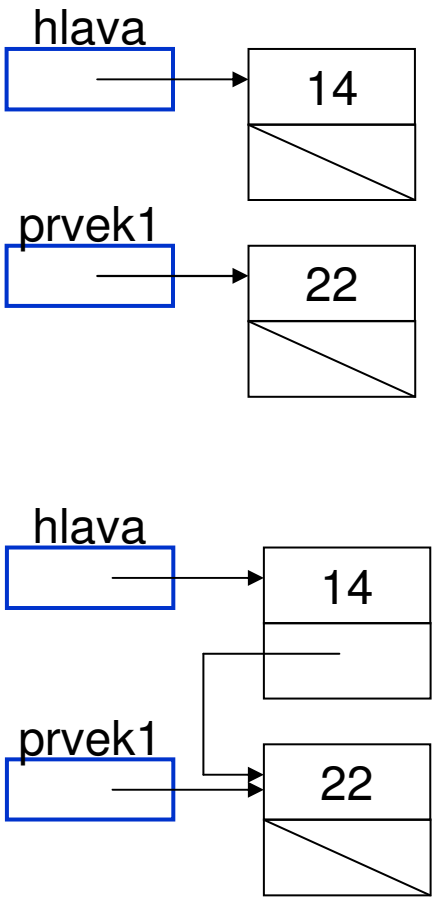
Spojové seznamy II

hlava = new Prvek (14, null);

prvek1 = new Prvek (22, null);

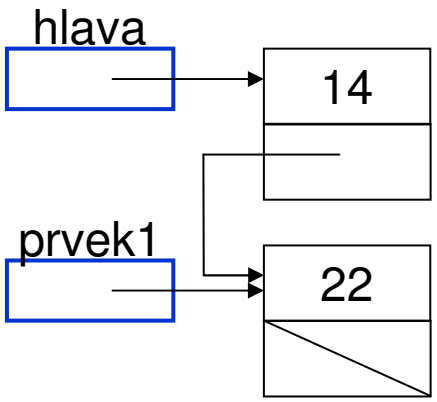
prvek1 = new Prvek (22, null);

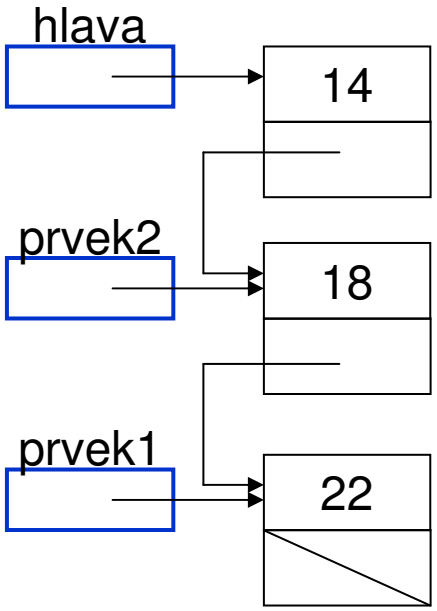
hlava = new Prvek (14, prvek1);



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);
```

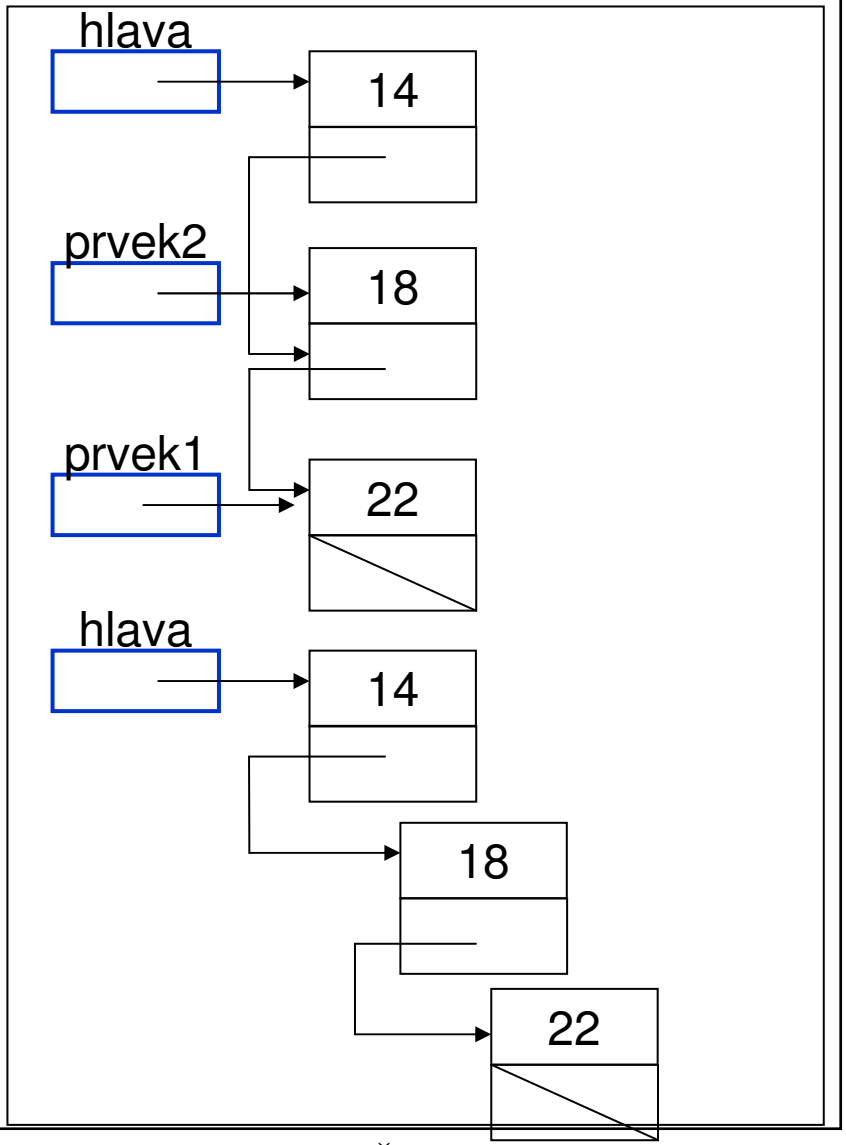




Spojové seznamy IV

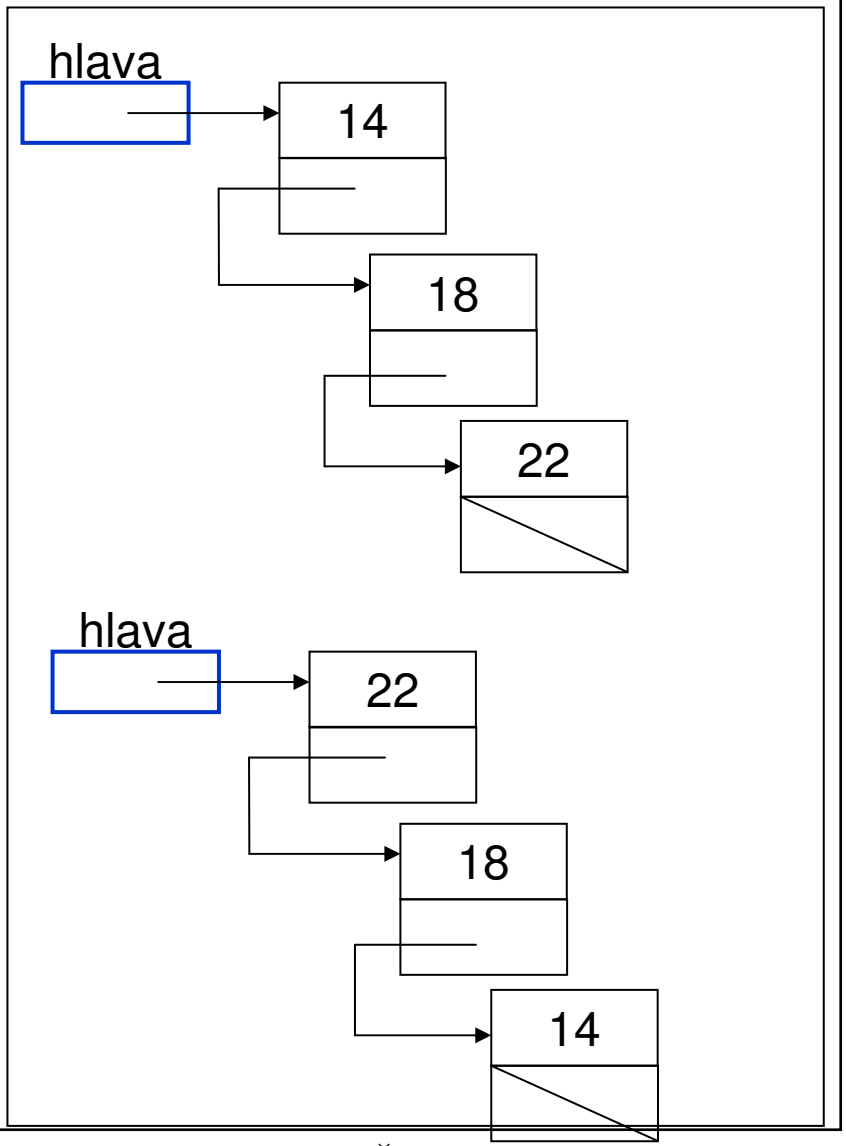
```
prvek1 = new Prvek (22, null);
prvek2 = new Prvek (18, prvek1);
hlava  = new Prvek (14, prvek2);
```

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



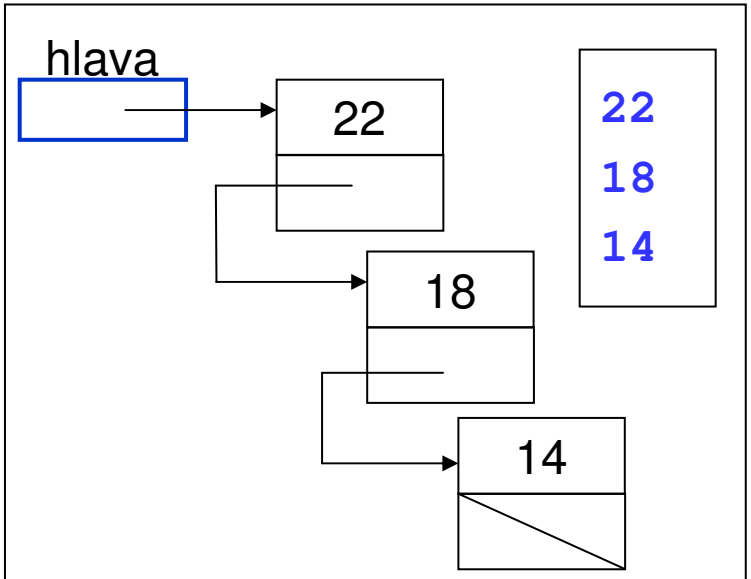
Spojové seznamy V

```
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;
}
```



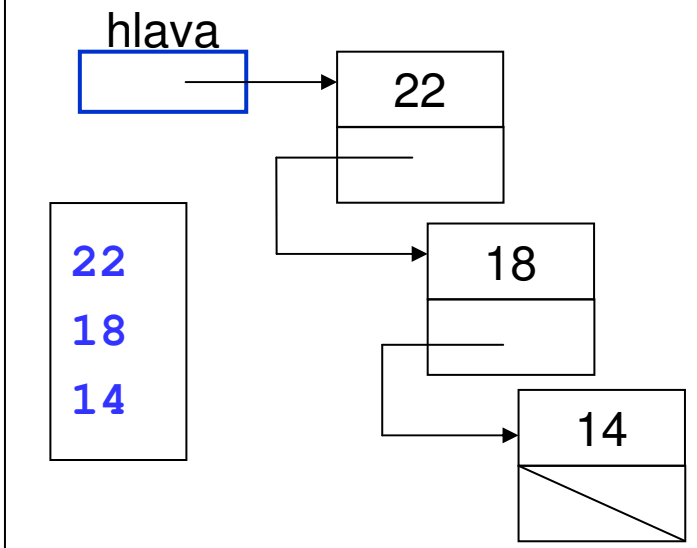
Spojové seznamy VI

```
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());
```



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();
}
```




```
22
18
14
```

```
22
18
14
```

Příklad použití spojového seznamu

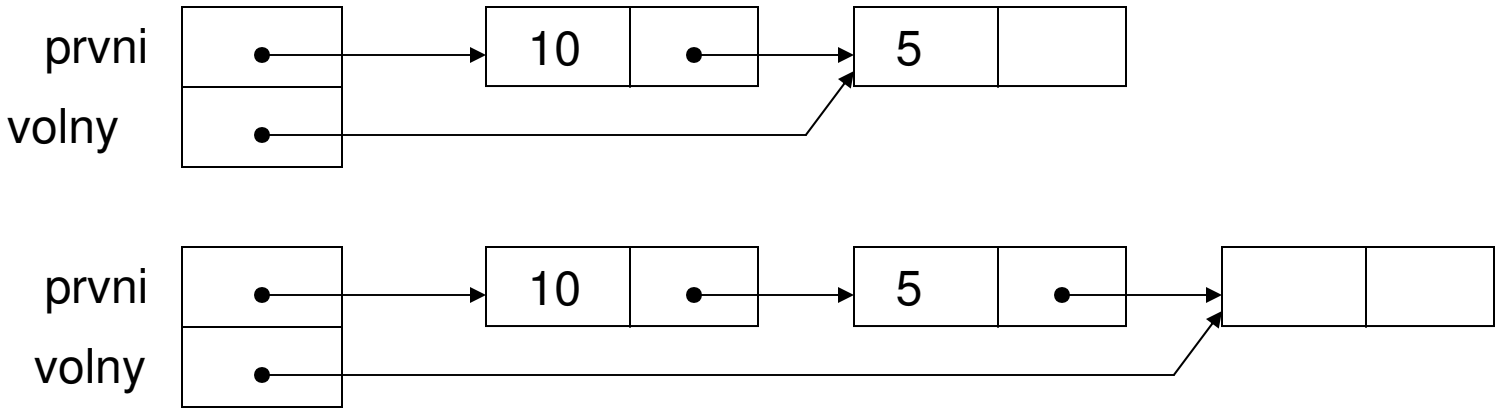
- Výsledné řešení:

```
public class ObraceniCisel {
   public static void main(String[] args) {
      System.out.println("zadejte řadu zakonč. nulou");
      Prvek prvni = null;
      int cislo = sc.nextInt();
      while (cislo!=0) {
         prvni = new Prvek(cislo, prvni);
         cislo = sc.nextInt();
      }

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

V jakém pořadí se čísla vypíšou?

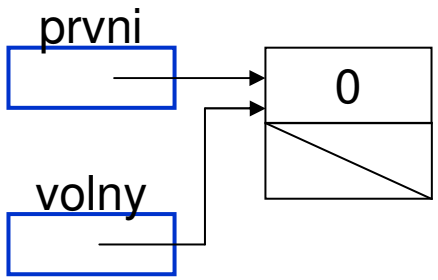
Další operace se spojovým seznamem

- 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 jednu z následujících reprezentací:



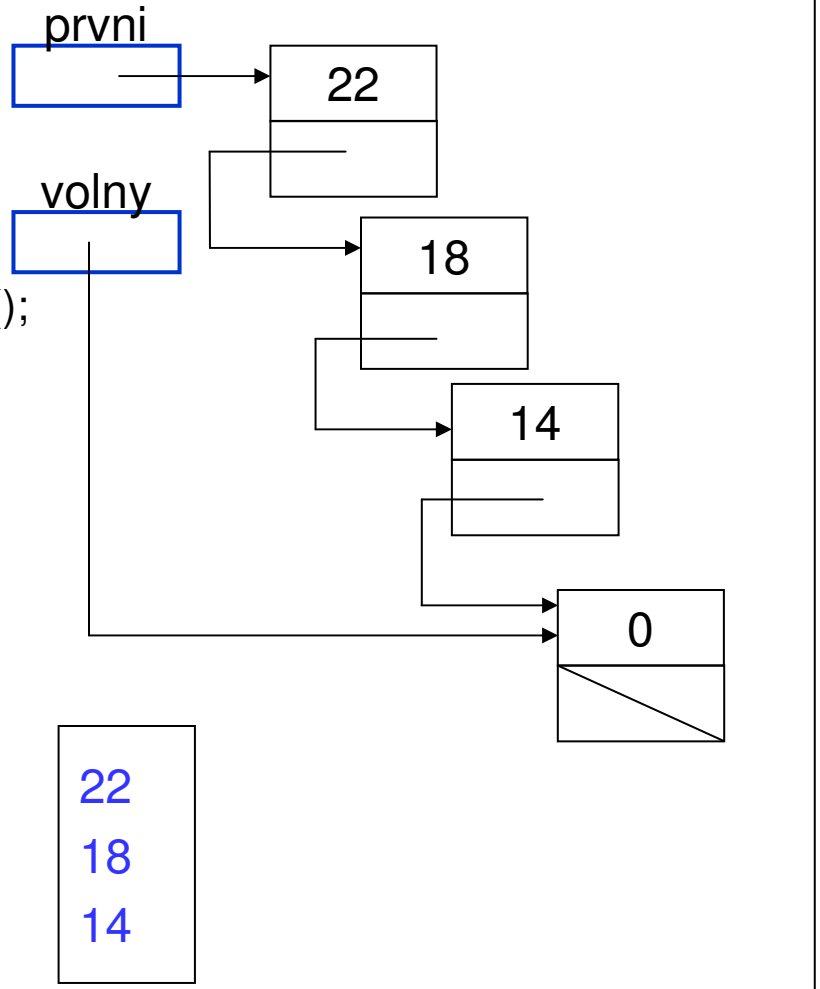
Spojové seznamy VII

```
Prvek volny;
Prvek prvni;

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



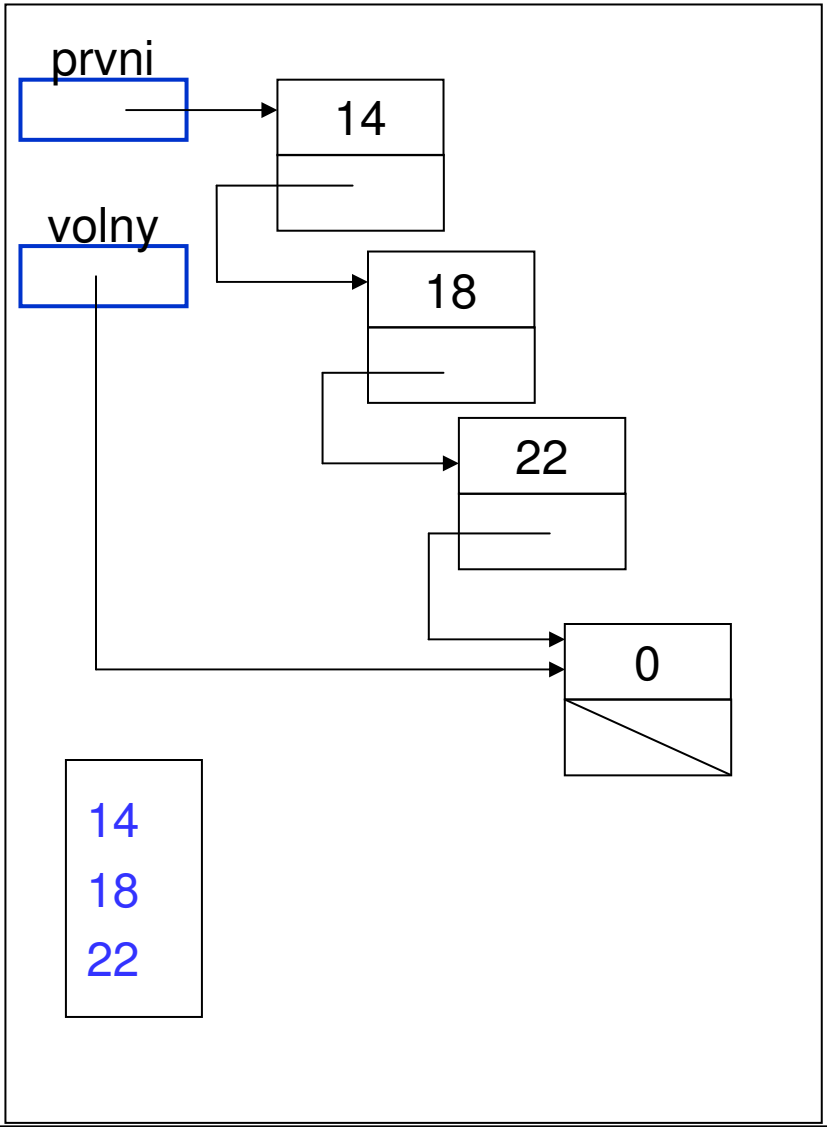
Spojové seznamy VII

```
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);
jednoduseK.vypis();

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



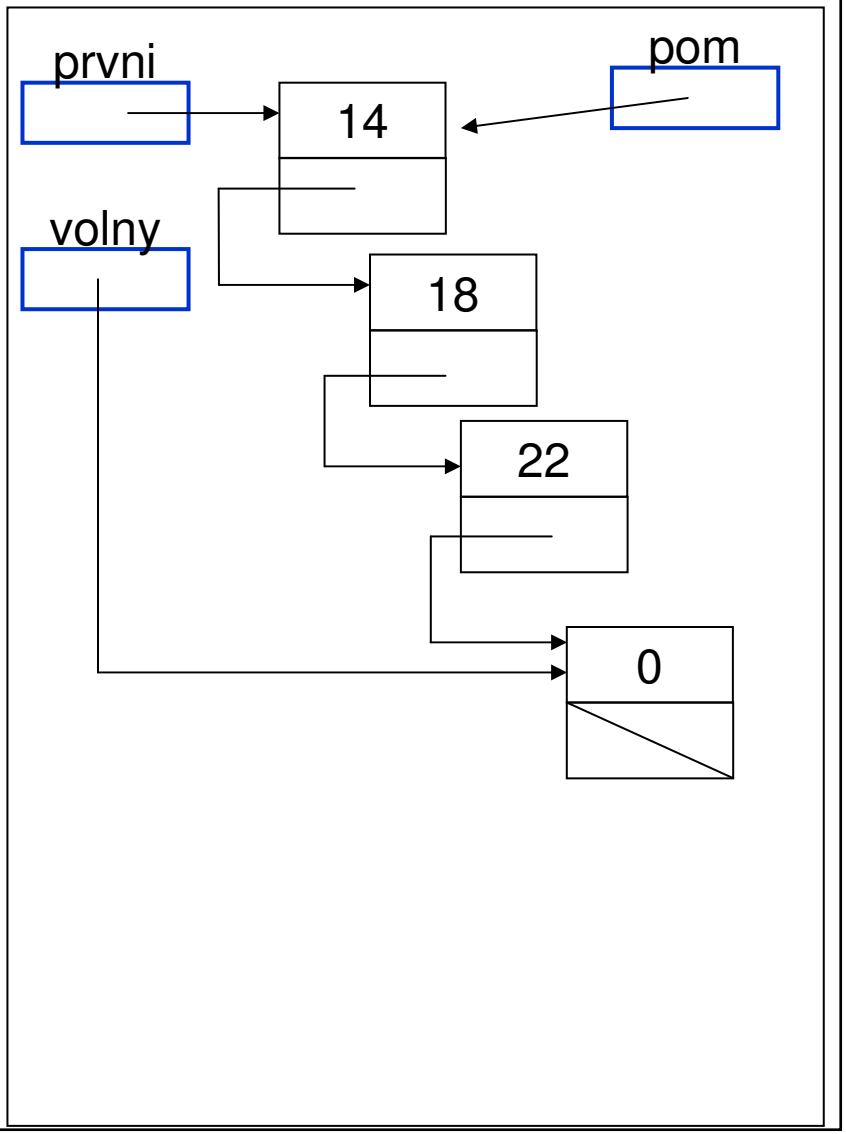
Spojové seznamy VIII

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


jednoduseZ.vlozNaZacatek(22);
jednoduseZ.vlozNaZacatek(18);
jednoduseZ.vlozNaZacatek(14);
jednoduseZ.vypis();
```



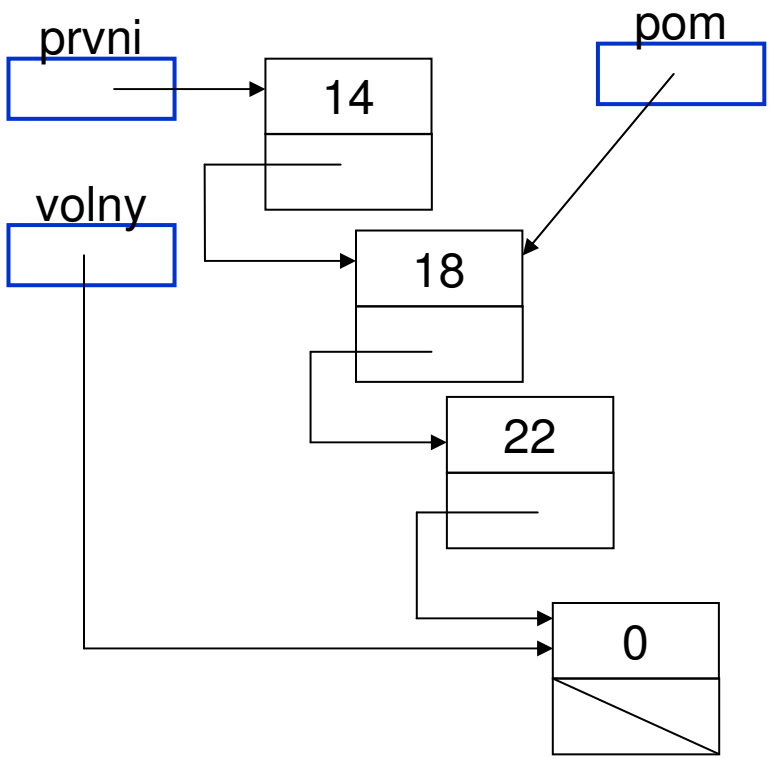
Spojové seznamy IX

```
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;
}
```



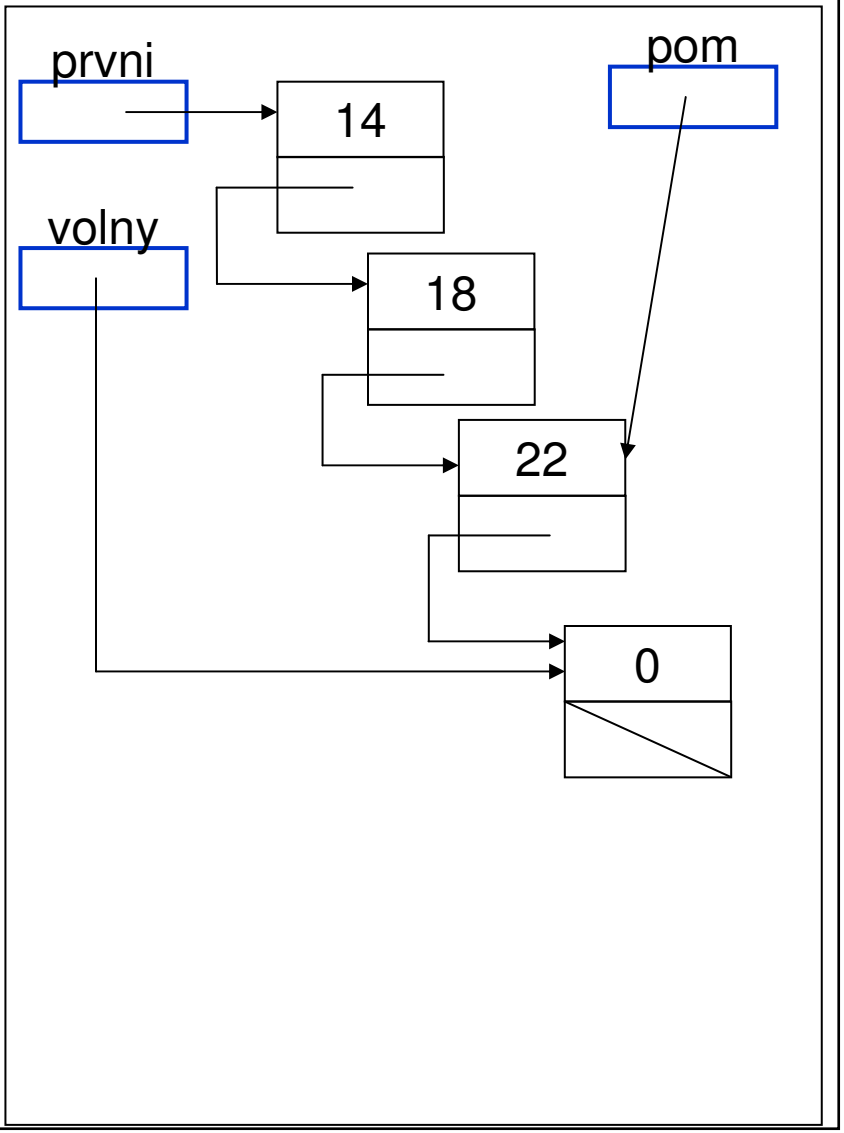
Spojové seznamy IX

```
public boolean jePrvkem1(int x) {

    Prvek pom = prvni;
    while (pom.hodn!=x && pom!=volny)
        pom = pom.dalsi;
    return pom!=volny;
}
```



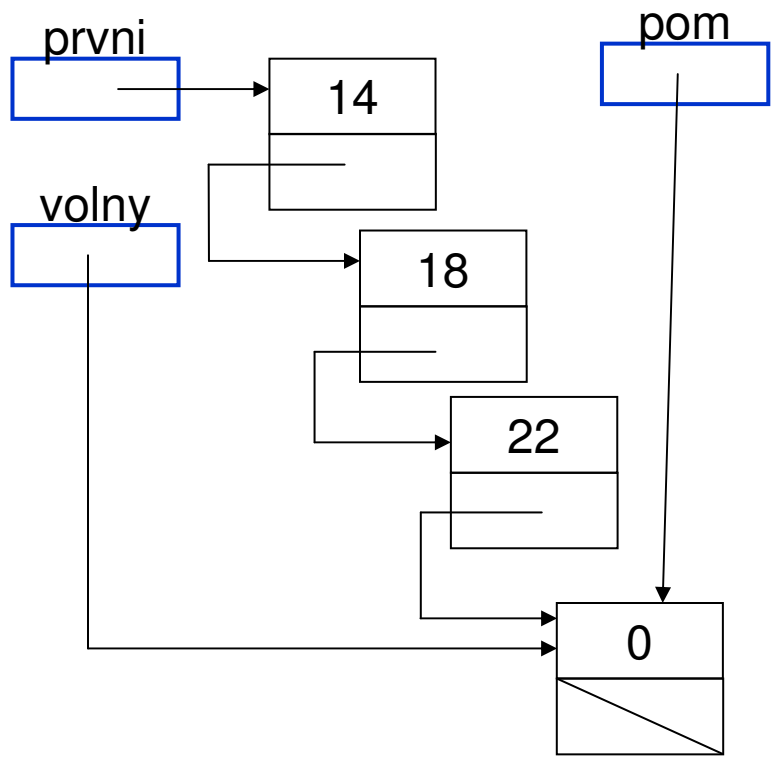
Spojové seznamy IX

```
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;
}
```



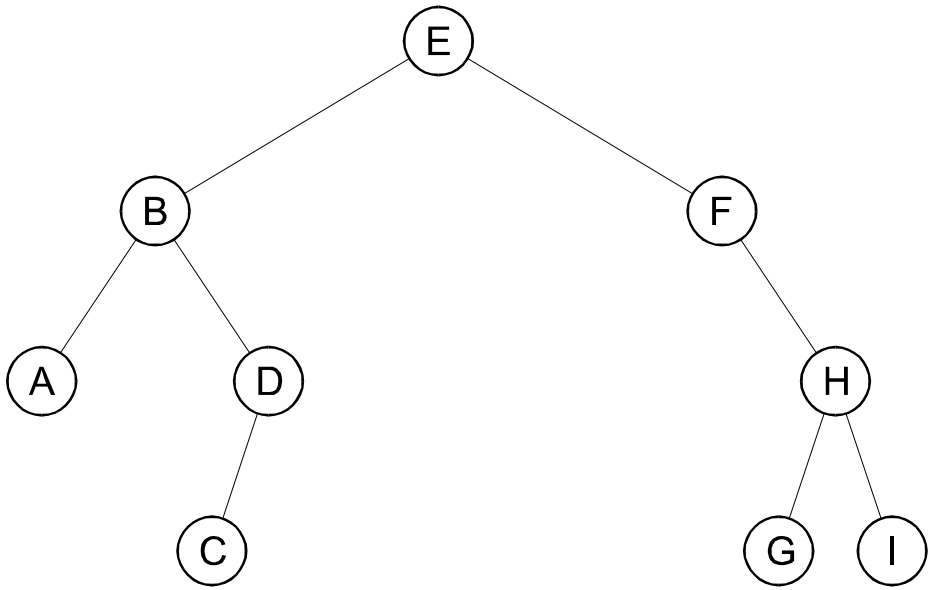
Použití třídy SeznamCisel

- Program, který přečte řadu čísel zakončených nulou a vypíše:
  - přečtená čísla v opačném pořadí
  - seznam všech unikátních (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/potomků
- Binární strom: každý prvek (uzel) má nanejvýš dva následníky



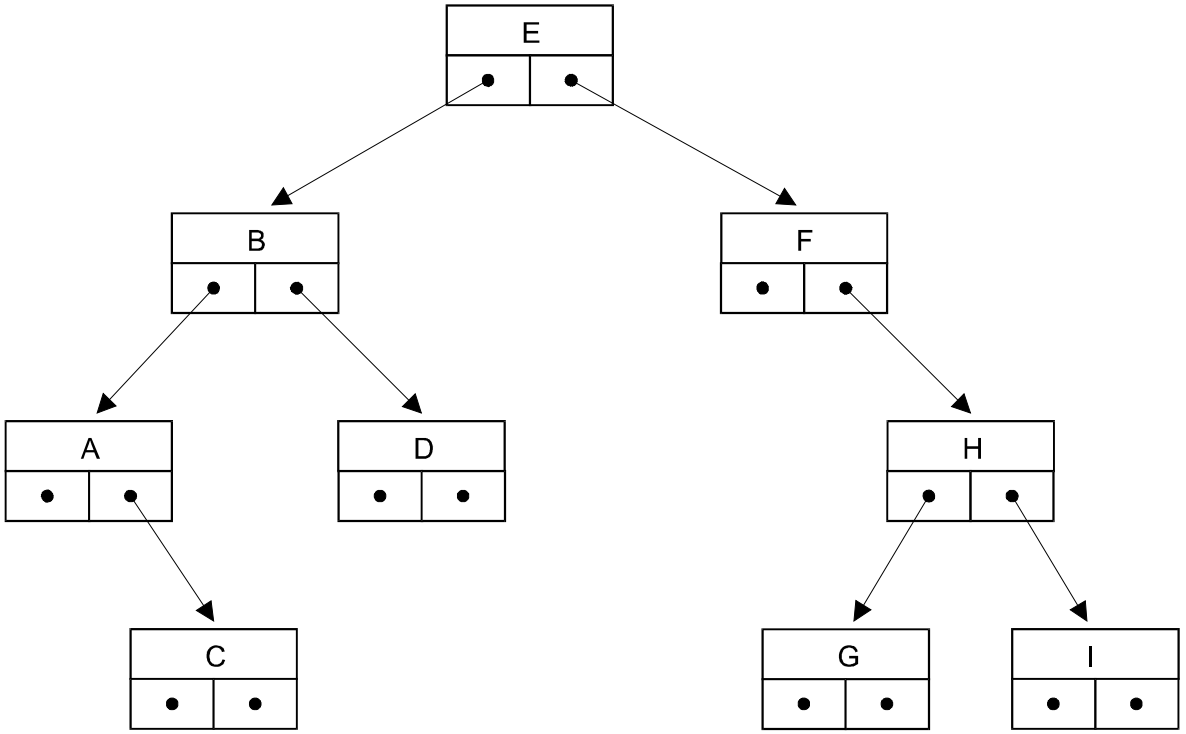
- Některé pojmy: kořen stromu, levý podstrom, pravý podstrom, list

Realizace binárního stromu

- Třída pro realizaci:

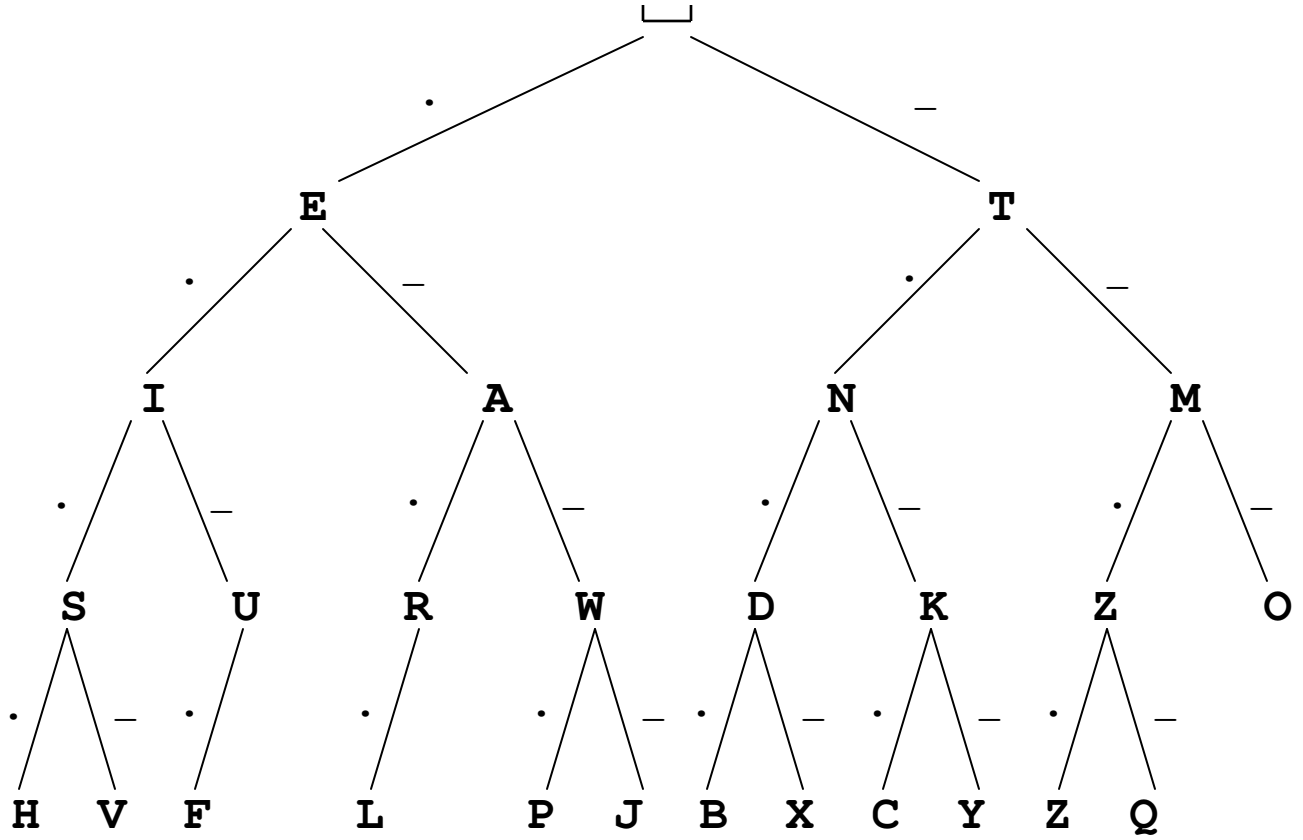
```
class Uzel {
  char hodn;
  Uzel levy, pravy;
  ...
```

- Příklad binárního stromu:



Příklad – 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("")) {
      Sys.pln(dekoduj(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 ab.

```
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; // včetně mezery
          aktualni = koren;
        }
      else {
        vysl = vysl + '?';
        aktualni = koren;
      }
    }
    return vysl;
  }
```

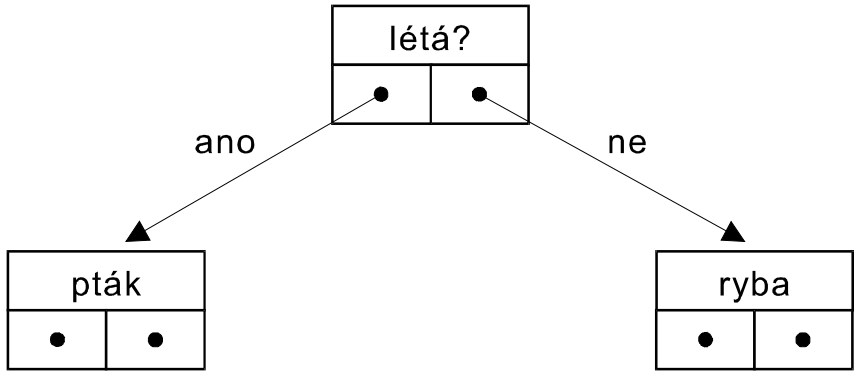
Příklad - hra „Jaké zvíře si myslíš“

- Příklad dialogu:

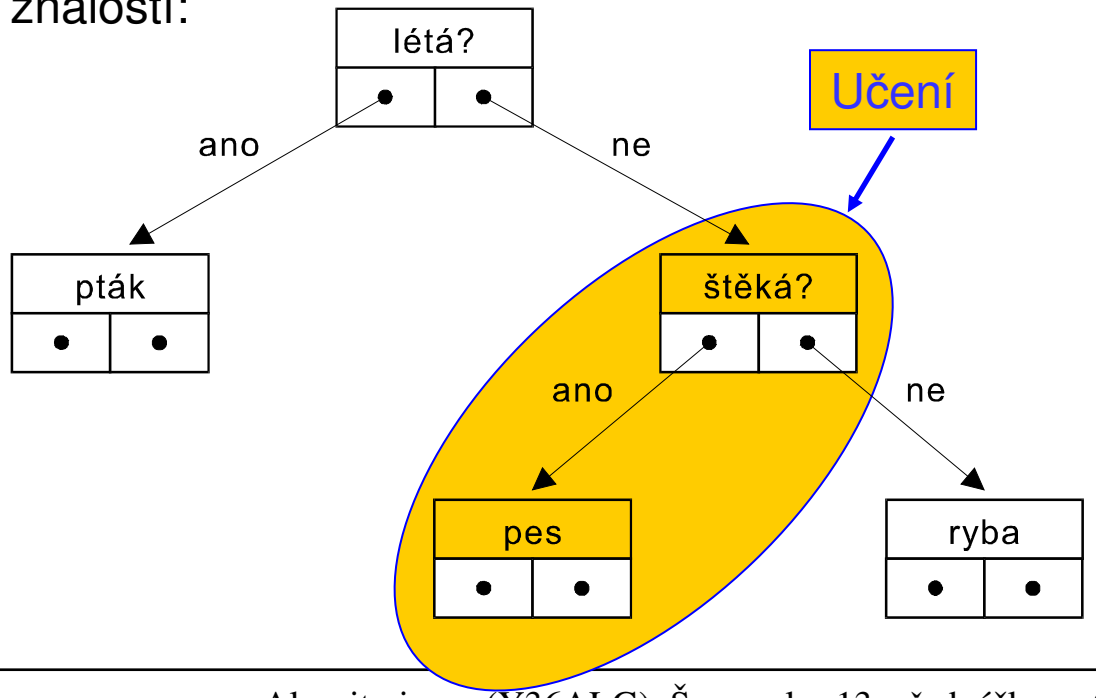
```
Myslíte si nějaké zvíře?
Ano

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 akt.uzlu";
if ("odpověď je ano")
   "hádání bylo úspěšné"
else {
   "hádání bylo neúspěšné";
   "doplň znalosti“;
}
```

Příklad - hra „Jaké zvíře si myslíš“

- Podrobné řešení
- Třída uzlů stromu:

```
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“

- Pomocné funkce:

```
static void doplnPodstrom(Uzel p) {
  String noveZvire, novaOtazka;
  Uzel novyAno, novyNe;
  System.out.println("Jaké zviře jste myslel?");
  noveZvire = sc.nextLine();
  System.out.println("Napište otázku"+
     "vystihující rozdíl mezi "+ noveZvire+" a "+p.text);
  novaOtazka = sc.nextLine();
  System.out.println("Pro zvíře, které jste myslel," +
                                  "je odpoveď 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;
}
```

Tolik přednášky z Algoritmizace Y36ALG

- Co zbývá:
  - zápočet
  - pokud máte nárok na 1 ze semestru, jen zápis zkoušejícím bez zápisu na zkoušku
  - zkouška
- Zkouškové termíny:
  řádné 15.1.2010, 22.1.2010
- Přeji úspěšné zkouškové období