Rekurze a složitost

Faktoriál

- Iterativní verze

```
public static int faktorial(int cislo){
    int vysledek  = 1;
    for(int i = 2; i <= cislo; i++){
        vysledek *= i;
    }
    return vysledek;
}
```

- Rekurzivní verze

```
public static int faktorialR(int cislo){
    if(cislo == 1 || cislo == 0){
        return 1;
    }
    return cislo * faktorialR(cislo - 1);
}
```

Fibonacciho čísla

- Řada 0 1 1 2 3 5 8 13 21 ...
- definována jako F(i) = F(i-1) + F(i-2)
- F(0) = 0
- F(1) = 1

Fibonacciho čísla II

- Rekurzivní řešení

```
public static int fibR(int cislo){
    if(cislo == 0) return 0;
    if(cislo == 1) return 1;
    return fibR(cislo - 1) + fibR(cislo - 2);
}
```

- Iterativní řešení

```
public static int fib(int cislo){
    int a = 1, b = 0;
    if(cislo == 0) return 0;
    if(cislo == 1) return 1;
    for(int i = 2; i <= cislo; i++){
        int p = a + b;
        b = a;
        a = p;
    }
    return a;
}
```

Hanojské věže

- Rozložení problému na podproblémy
- Ideální pro využití rekurze

```
public static void hanoj(int kolik, int odkud, int kam, int
pres){
    if(kolik > 1) {
        hanoj(kolik-1,odkud,pres,kam);
    }
    System.out.println("Presun disk ze sloupu "+odkud+" na
sloup "+kam);
    if(kolik > 1){
        hanoj(kolik-1,pres,kam,odkud);
    }
}
```