1. Domača naloga

Testiranje in kakovost

Kje smo sedaj?

- **Seznam.java**
  - **Javanski vmesnik** (interface) z abstrakcijo metod, ki so zahtevane v implementaciji podatkovne strukture
- **SeznamiUV.java**
  - **Uporabniški vmesnik** za delo z različnimi podatkovnimi strukturami
- **Sklad.java, PrioritetnaVrsta.java, Bst.java**
  - Že implementirane **podatkovne strukture** (sklad, prioritetna vrsta in osnovno binarno iskalno drevo).
- **Testi enot in integracijski testi**
  - Popolno napisani vsi testi
  - Vtičnik za pokritost programske kode – 100% pokritost

Specifikacija – 1. del

- Dopolnite vmesnik **Seznam** z metodo **List<Tip> asList(),** ki vrne seznam z elementi podatkovne strukture
- Ustrezno dopolnite vse podatkovne strukture **(Sklad, Bst** in **PrioritetnaVrsta**) z novo metodo. Metoda **asList()** naj vrne elemente v naslednjem vrstnem redu:
  - **Sklad** – od vrha proti dnu
  - **PrioritetnaVrsta** – nivojski izpis kopice
  - **Bst** – vmesni vrstni red (inorder)
- Napišite tudi vse ustrezne teste enot za to metodo, da bo pokritost kode s testi povsod 100%

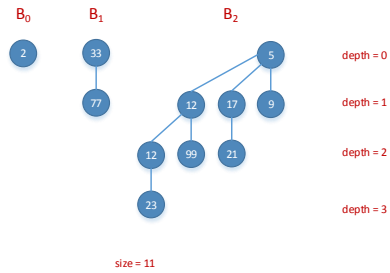
Specifikacija – 2. del

- Realizirajte novo podatkovno strukturo **Binomska kopica*** (*BinomskaKopica.java*), ki implementira vmesnik Seznam
  - Struktura mora uporabljati generični podatkovni tip, ki je vsaj **Comparable**
  - realizirati je treba **MAKSIMALNO** binomsko kopico
  - Vse metode se izvedejo na ustrezen način glede na definicijo binomske kopice
  - Metoda **asList()** naj elemente binomske kopice vrne v zaporedju, ki ga določa obratni vrstni red izpisa (postorder) binomskih dreves po vrsti (od najmanjšega – levega do največjega – desnega)
- Za binomsko kopico je treba implementirati vse **teste enote** v testnem razredu ***BinomskaKopico23Test***
  - Poskrbite za 100% pokritost razreda

* glej T. Cormen, C. Leiserson, R. Rivest: Introduction to Algorithms (str. 400 - 419)

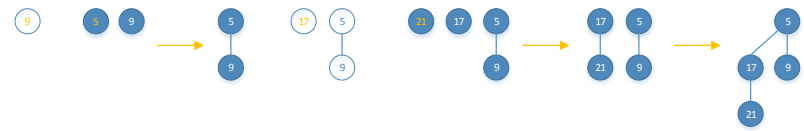
Binomska kopica

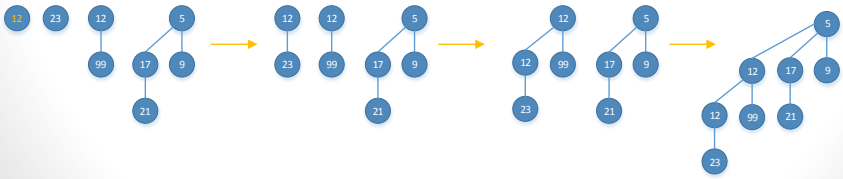
- Zaporedje binomskih dreves, ki zadoščajo pogojem
  - vsako drevo je min-urejena kopica (v dom.nal je treba narediti max)
  - imamo bodisi 0 bodisi 1 binomsko drevo reda *k*
- `depth()` – globina najvišjega binomskega drevesa
- `size()` – število elementov v celotni binomski kopici
  - velikost posameznih dreves = 2^{depth}



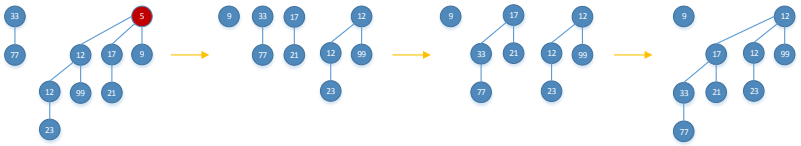
Dodajanje v binomsko kopico

- Primer minimalne kopice (v dom.nal je treba realizirati maksimalno kopico)
  - 9, 5, 17, 21, (99,12,23), 12

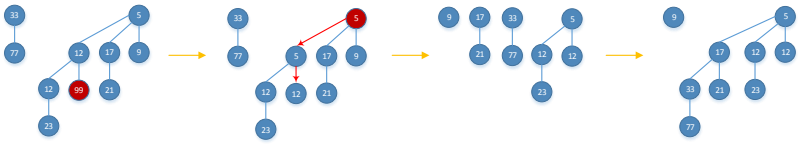




Brisanje min elementa



Brisanje poljubnega elementa



Implementacija

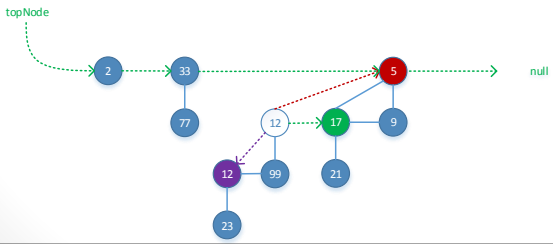
Ena od možnih implementacij:

```
public class BinomialHeap<Tip extends Comparable> implements Seznam<Tip> {

  class BinomialHeapNode {
            private Tip key;
            private int degree;
            private BinomialHeapNode<Tip> parent;
            private BinomialHeapNode<Tip> child;
            private BinomialHeapNode<Tip> sibling;

            ...
  }

  private BinomialHeapNode<Tip> topNode;
```



Specifikacija – 3. del

- Posodobite uporabniški vmesnik **SeznamiUV** z novimi ukazi za delo z vašo podatkovno strukturo:
  - dodajte nov ukaz **asList**, ki za izbrano podatkovno strukturo izpiše seznam z njeno vsebino
  - za preklop na uporabo binarne kopice dodajte **use bk**
- Skratka vzpostavite stanje, ko lahko nad vsako izmed štirih podatkovnih struktur izvajamo vseh deset podprtih ukazov: **add, removeFirst, getFirst, size, depth, isEmpty, reset, exists, remove** in **asList**
- Implementirajte vse ustrezne **integracijske teste** za novo strukturo in ukaze
  - Teste dodajte v *SeznamiUVTest.java (enako kot za ostale podatkovne strukture)*

Specifikacija – 4. del

- Delo po principih **osebnega razvojnega procesa** (PSP)
- Zahtevano je vodenje naslednjih beležk:
  - Beležka porabe časa
  - Beležka opravil
  - Beležka napak
  - Projektni plan (v obsegu, kot je predstavljen na vajah)

Izvedba

- **Ustrezna izvedba binomske kopice**, kjer so striktno upoštevana pravila (Cormen...)
- Razred **BinomskaKopica** mora vsebovati vse metode vmesnika **Seznam**:
  - **void add (Tip e);** -> doda element e v kopico
    - Dodajanje duplikata je dovoljeno
    - Tudi struktura brez elementov je bin. kopica
  - **Tip removeFirst();** -> odstrani in vrne max element
    - Dejansko odstani največji koren v enem drevesu, sledi popravljanje
  - **Tip getFirst();** -> vrne max element
    - Vrne največjega izmed korenov dreves kopice
  - **int size();** -> vrne št. elementov v kopici
    - Štejemo sproti ali posebna metoda

Izvedba

- **int depth();** -> vrne globino največje kopice
  - lahko kar stopnja zadnjega bin. drevesa
- **boolean isEmpty();** -> vrne true, če je kopica prazna
- **boolean exists(Tip e);** -> vrne true, če je element prisoten
  - Iskanje je problem
  - Smiselno pregledovanje posameznih dreves -kopic (optimizacija)
- **Tip remove(Tip e);** -> odstrani in vrne poljuben element
  - Lahko posredno z metodo za zmanjšanje vrednosti:...
- **List<Tip> asList()** -> vrne seznam z vsemi elementi kopice
  - Vrstni red elementov določa obratni obhod (postorder) vseh binomskih dreves po vrsti od najmanjšega do največjega
  - **List** -> **ArrayList**

Poleg razreda z implementacijo zgornjih metod ustvarite še notranji razred **BinomskaKopicaNode**

Ključna vloga pomožne metode za popravljanje bin.kopice

Kaj oddati?

- Celoten projekt in ustrezne beležke (PSP) v eni zip datoteki
- Struktura ZIP datoteke (*tk_dn1_vpisnaSt.zip*)
  - **Koda.zip**
    - Ustvarite jo z izvozom projekta v zip datoteko (File -> Export Project -> To ZIP...)
  - **PSP.zip**
    - *poraba_casa.xlsx (ali .pdf)*
    - *belezka_opravil.xlsx (ali .pdf)*
    - *belezka_napak.xlsx (ali .pdf)*
    - *projektni_plan.xlsx (ali .pdf)*

Rok za oddajo

- Rok za oddajo je **nedelja, 13. maj 2018 ob 23.55**
- Naslednje vaje (25.in 26. 4.) bodo namenjene:
  - predstavitvi izpolnjevanja PSP Projektnega plana
  - pomoči pri reševanju 1. domače naloge
- Zaradi praznikov v tednu 30.4 - 3.5. ne bo vaj (bodo pa 3.5. predavanja!)
- **Po oddaji bo na naslednjih vajah zagovor domače naloge.**