Domaći zadatak

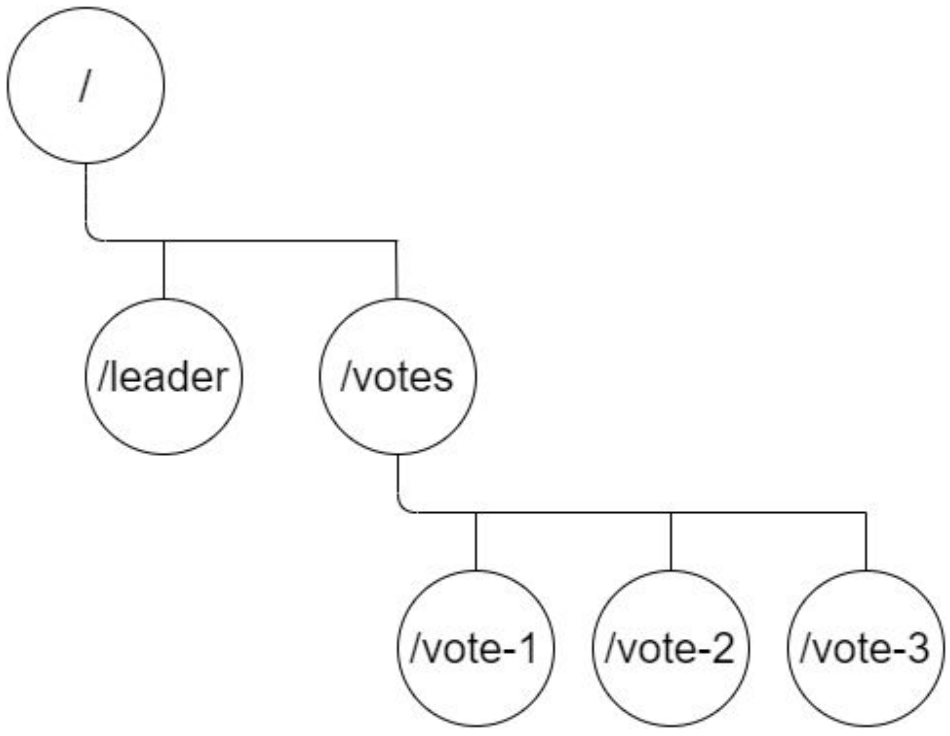
Zookeeper

Napraviti Java klasu koja će pomoću ZooKeepera simulirati glasanje. Cilj je pokrenuti nekoliko istih procesa koji će među sobom odrediti **leader**-a. Leader će započeti glasanje i prikupiti i objaviti rezultate. Ostale instance klase će biti **follower**-i. Svaka od njih daće svoj glas, i kada svi završe glasanje leader će objaviti konačni rezultat, a svaki od followera će objaviti da li je zadovoljan rezultatom ili ne.

Leader Election

Leader se određuje tako što program pri registraciji pokušava da kreira ephemeral **/leader** čvor. Onaj koji uspe biće **leader** do kraja glasanja a svi ostali koji nisu uspeli biće **followers**.

Napomena: Pošto radimo u simuliranim uslovima ovde nećemo paziti na to da li je leader prekinut i njegov čvor obrisan. Razmisliti kako bi se ovaj problem rešio.



Leader

Instanca klase odeđena za leadera pokreće glasanje tako što uspešno kreira **/leader** čvor i pri kreiranju u njega upisuje string **start**. Prikuplja glasove koji će pristizati kao deca čvora **/votes**. Taj čvor možete kreirati i iz komandne linije i uvek koristiti isti, važno je da bude **PERSISTENT** da bi mogao da ima decu. Postaviti watch i čekati da se skupe glasovi. Svaki glas predstavljen je kao **EPHEMERAL_SEQUENTIAL** čvor koji je dete čvora **/votes** a vrednost čvor je jedna od 2 opcije "**yes**" ili "**no**" (**1** ili **0**, **da** ili **ne**).

Kada se skupi dovoljan broj glasova leader čita svaki od njih i broji kojih ima više. Ispisuje rezultat na ekran i objavljuje rezultat glasanja preko Zookeepera tako što menja vrednost svog leader čvora na rezultat glasanja.

Koristiti **setData(String path, byte[] data, int version)** za promenu vrednosti **/leader** čvora. Ovoj metodi moramo proslediti **version** kao kod delete ali će nam on uvek biti 0 jer je to prva promena.

Follower

Pri pokretanju pročitati vrednost **/leader** čvora i postaviti watch za kasnije. Ako je pročitana vrednost **start,** započeti glasanje.

Follower daje svoj glas tako što kreira **EPHEMERAL_SEQUENTIAL** čvor **/votes/vote-0000000xx** u kome je upisana vrednost glasa. Glas može biti **da** ili **ne** i odrediti ga pomoću klase Random() kao jedan od brojeva 0 i 1.

Kada je glas poslat, follower pamti kako je glasao i čeka da se glasanje završi tako što čeka promenu vrednosti na leader čvoru. Kada leader objavi rezultat pročitati vrednost i ispisuje na ekran svoj **id** i poruku da li je njegov glas pobedio ili izgubio.

```
FINAL RESULT: no
55a450ee: Moj glas je izgubio...
167fedee: Moj glas je pobedio!
690d9009: Moj glas je pobedio!
```

Testiranje

U main metodi kreirati i pokrenuti 4 instance iste klase koje će se automatski dogovoriti koja će biti leader, a koje 3 followers.

Leader može smatrati da je glasanje završeno tj da su svi glasali u trenutku kada **/votes** čvor ima 3 deteta.

Pretpostaviti da ćemo uvek raditi sa 4 instance i smatrati da se njihovi čvorovi neće obrisati tokom izvršavanja aplikacije. Ephemeral nisu neophodni ali ih možemo koristiti radi jednostavnijeg testiranja.

Nije potrebno obezbediti se i hendlovati tipove otkaza koji nisu pomenuti u ovom uputstvu ali razmisliti gde može doći do potencijalnih propusta i kako bi bilo moguće ispraviti ih.

Primer main metode:

```
public static void main(String[] args) throws Exception {
    Logger.getRootLogger().setLevel(Level.OFF);
    List<Domaci> domaci = new ArrayList<>();

    for (int i = 0; i < 4; i++) {
        domaci.add(new Domaci());
    }

    for (Domaci d : domaci) {
        d.start();
        d.register();
    }

    Thread.sleep( millis: 120000);

}
```