Readme,

För att köra, kan ni behöva lägga till JSON-simple biblioteket i ert program. Jag spara jar filen i den komprimerade mappen. Se *json-simple-1.1.1.*

Programmets funktion i text,

1. *Main*, instantierar sig själv och anropar methoden *run*();
2. *Run* skapar anropar *getStops* i *ApiConnection*,
3. *In ApiConnection.getStops();*
    a. *checkURLConn* kollar om apin är redo för en uppkoppling, om svaret är 200 returnerar metoden true, annars false,
    b. Om true*,*
        i. *getJsonArray*() tar emot en inputstreamreader och använder sig av bufferedstreamreader tillsammans med JSONParser i JSON-simple biblioteket,
        ii. metoden returnerar en JSON array med de objekt vi är intresserade av. Retur datat läggs i arrayen ”*data”*
        iii. En ”*Enhanced*” forloop iterar över *data*, konverterar en specifik plats ur *data* och sparar id för en specifik hållplats med id som key, string som value i en HashMap.
        iv. *getStops()* returnerar HashMappen till Main.Run().
    c. *Om falsk*
        i. ”Throw RunTime error”
4. Main.Run() skapar en ny instans av Busslines och anropar *busslines.getBussLines()*
    a. Stegen i 3.a upprepas,
        i. Upprepar 3bi,
        ii. Upprepar 3bii,
        iii. datastrukturen följer mönstret,
        [line no1, hållplats id1, ….]
        [line no1, hållplats id2, ….]
        [line no1, hållplats id3, ….]
        [line no2, hållplats id11, ….]
        iv. För att ta ut datat på ett effektivt sätt så har jag skapat 2 forloopar,
            1. En yttre forloop som iterar över alla objekt, (int i = 0; ; i++)
            2. En inre som börjar på i och räknar, hur många objekt framåt som tillhör samma busslinje, den inre forloppen ligger i metoden *countStops().* Metoden *countStops()* adderar objekt till sin array för att sedan returnera en count som beskriver hur många steg före den gått.
            3. För att den yttre inte ska iterera över samma objekt som den inre sätts i slutet av loopen i +=count. Det blir matematiskt korrekt eftersom att forloopen också adderar 1 för varje varv. Ex den yttre forloopen står på objekt 1, den inre har gått 5 objekt framåt dvs till objekt 6. 1+5=6 där vi redan är, men för nytt varv körs i++ dvs i = 7;
        v. Ett nytt objekt skapas Buss, som innehåller -String:ID & -arrayList<String>:busstops kallas *bussline*

vi. I slutet av varje iteration i den yttre forloopen så anropas *TopRank.addToRank(bussline)*.
vii. *TopRank.addToRank(bussline)* innehåller en lista med 10 bussar, lägger till nummer 11, sorterar listan på antalet stops från högst till lägst och tar bort det sista objektet.

b. Main.run() anropar sedan metoden ”*busslines.printTopScorers()*” som skriver ut listan.

Antaganden som jag gjort,

- Bussar åker t/r, jag kan tänka mig att inte alla bussar åker exakt samma väg t/r, för att få med alla busshållplatser så behövs t/r, detta går dock enkelt att styra eftersom i bussobjektet med hållplatser finns directionCode som antingen är 1 eller 2.

Val

- Klassen Buss är en inre privat klass. Det ökar säkerheten, förbättrar prestationen och i min mening förbättrar kodöversikten.
- TopRank sparar sorterar bara 11 objekt. Det hade tagit att sortera en lista som blir 1000tals objekt flera gånger, det är betydligt ”billigare” att sortera en lista med bara 11 objekt flera gånger.

Prestation,

- Tid från start till slut ofta ca 2s-5s inkl nedladdningstid etc.
- Jag har försökt köra en loop som kör 10 varv och räknar ut medeltiden, men då säger hemsidan ifrån. DDOS skydd?

Förslag på förbättringar,

- Jag har letat sätt att filtrerar ännu mer i själva apianropet. Jag har inte hittat att det går, men skulle det gå hade det gått att effektivisera mycket mer.
- Vissa hållplatser skrivs ut flera gånger, jag förstod inte varför först men när jag analyserarade värdena i api, då är det samma hållplats flera gånger, både till idnummer och namn till samma nummer.
    - Jag har säkertställt detta på två sätt;
        - Kolla längst ner i Busslines, kopiera in mot ett anrop i webbläsaren och sök hur många träffar ni får på respektive kod, de kommer stämma med de till höger. Siffran 232 ex kan hämtas från bussStops.size();
        - Jag har kollat manuellt i api och sett att både namn & idnummer kommer flera gånger för samma linje.
    - En möjlig lösning som skulle lösa problemet är att också hämta stopArea i ett nytt apianrop. Det hade jag gjort om antalet api anrop inte överskridit maxtalet
- Skriva en specialanpassad hash & compare funktion, detta skulle kunna förbättra prestationen.
- Jag tänkte börja med Junit tester + utförligare performance tester men antalet apianrop är slut.