Programmet kjøres på vanlig måte, ved kommando *java Oblig2*.

Alle klasser (og main) ligger i denne fila.

Oppgaven er løst ved bruk av Map for å representere grafen.

- 1 Map som inneholder alle skuespillere, der nøkkel er skuespiller- ID og verdi er skuespiller- objekt (Actor)

- 1 Map som inneholder alle filmer, der nøkkel er film- ID og verdi er film- objekt (Movie)

- 1 Map som inneholder filmer og skuespillere, der nøkkel er film, og verdi er alle skuespillerne som har spilt i denne filmen.

I tillegg inneholder alle skuespiller- objekter Map med alle filmer de har spilt i.

Det hadde vært mer ryddig å la alle film- objekter inneholde skuespillerne som hadde spilt i filmen, i stedet for å bruke 1 Map som er tilgjengelig for hele programmet. Gjorde det på denne måten fordi det virket som det gikk fortere, ettersom denne Map- ingen kunne lages samtidig med innlesing av skuespillere.

Bruken av 3 x static Map er også noe det kanskje hadde vært greit å unngå, og heller sendt disse med som parametere til funksjonene.

Kjøretiden er som følger:

V = skuespillere
E = filmer

| | | |
|---|---|---|
| readMovies() | Innlesing av movies.tsv | **O(E)** |
| readNodes() | Innlesing av actors.tsv | **O(V*E)** |
| countNodes() | Telling av antall noder i grafen | **O(1)** |
| countEdges() | Telling av antall kanter i grafen | **O(V*E)** |
| findWay() + printWay() | Leting etter korteste vei (uvektet) fra 1 node til 1 annen | O(V) + O(V+E) + O(V) |
| findChillWay() + printChillWay() | Leting etter korteste vei (vektet) fra 1 node til 1 annen | O(V) + O(E*log(V)) + O(V) |
| findComponents() | Leting etter antall komponenter og størrelsen på disse | O(V*E) + O(V*E) |
| Totalt | | O(V*E)*4 + O(V)*4 + O(E) + O(E*log(V)) + O(V + E)<br><br>= O(V*E) + O(V) + O(E) + O(E*log(V)) + O(V + E)<br><br>= O(V*E) + O(V + E) + E*log(V) |