INDU-Rapport

David Sermoneta

Mars 2023

Jag har valt att ta mig an projektet Tåg. När programmet sätts igång så bes användaren skriva in två '.txt' filer, en "stations"-fil och en "connections"-fil. Bifogat är 2 stycken fil-par som jag använt under skapandet av programmet. När dessa väljs ut så kommer `parse_through_file()` att gå igenom filerna och kontrollera att de är av rätt format, för att sedan returnera en lista av listor för alla rader respektive dokument. Till slut kontrolleras det även att dokumenten faktiskt är kompatibla. Om allt har gått bra så skapas stations-objekten genom att läsa listan från "stations"-filen m.h.a. `create_stations()`. Varje station har instans-attributen `name, delay, south, north`. Objekten skapas med endast namn och förseningssannolikhet. Direkt efter går vi igenom "connections"-listan m.h.a. `setup_connections()` som för varje lista i listan kallar på klass-metoden `add_connections()` i `Stations`-klassen, som ser till att södra stationer hamnar i `south`-instansen och norra stationer i `north`-instansen. Till slut bör alla stationer ha instanser som i följande exempel:

```
A.name #A ,
A.delay #0.02 ,
A.south #{'red' : 'B'} ,
A.north #{} .
```

Att `A.north` returnerar en tom dictionary svarar då mot att stations `A` är en slutstation i norra riktningen. När det är gjort och färdigt får användaren skriva in hur många tåg hen önskar simulera, och när det valet görs så sätts `create_trains()` funktionen igång, och den gör precis det man kan tänka sig att den gör, nämligen skapa tåg. Då skapas tåg-objekt med endast ett tal som namn, för att sedan bli tilldelade en slumpmässigt vald station m.h.a. modulen `random`, som sedan tar instansattributer från stationen såsom `current_station, direction, line, next_station`. Alla fås från stationen lottad i `create_trains()` m.h.a. klass-metoden `add_info()`.

Nu är det bara simulationen kvar, och när den körs så uppdateras helt enkelt bara tåg-objektets (givet att den passerar delay-checken) `current_station` med dess `next_station` och, om den hamnat i en slutstation, `direction`.

Jag har strukturerat största delen av mitt program till att kretsa kring klasserna `Stations`, och `Trains`. Stations-klassen är framför allt till att organisera all information från stations, och connections -filerna, för att göra den lättåtkomlig för tågklassen senare. Tåg-klassen är den som sen når och hämtar information från stations-objekten för att skapa tåg-objekten, och för att köra igång simulationen.

För andra uppgiften, valde jag att använda mig av en djupet-först sökning. Sökningen går till via rekursiva metoden `reach_from()` som sparar alla nårbara stationer från start till uppslagstabellen `memory` med värdet som svarar mot minsta antalet steg från start. Till slut kontrolleras det om den valda destinationen finns med i `memory` inom antalet steg som angavs av användaren.

Andra filen som jag lämnar in är ett enhetstest för att se kolla att de viktigaste delarna av programmet fungerar som de ska. Den exekverar nästan alla funktioner som ursprungliga programmet har, men utan någon som helst input från användaren. Filerna den använder sig av är `'s3.txt'` och `'c3.txt'`.