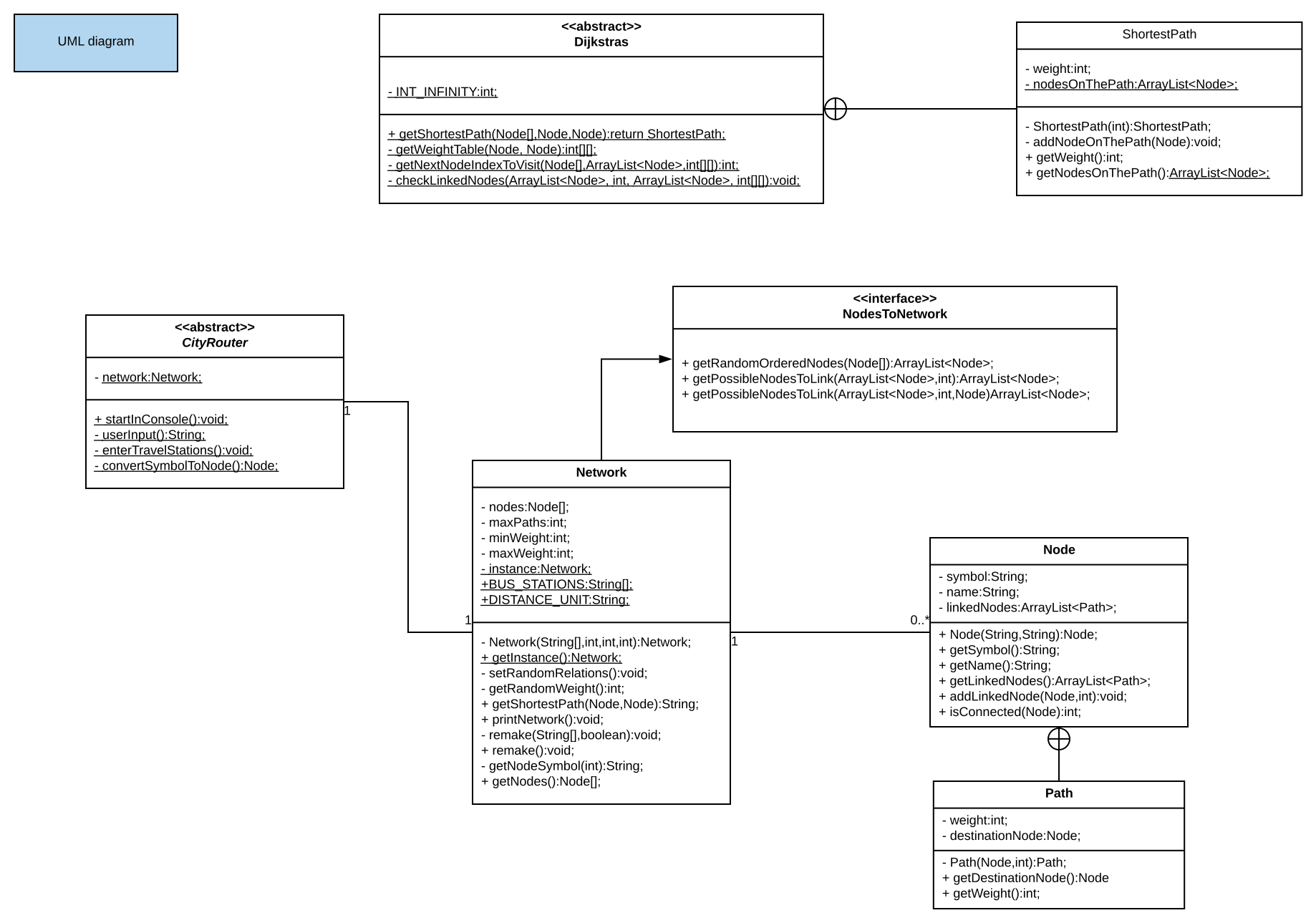
***Route city***

- Arkitekturen:  
Programmet består av en hub som heter CityRouter som har hand om menyn och användarinput.  
Den är abstrakt då man inte ska skapa objekt av den. Klassen som bygger upp nätverket av hållplatser heter Network och den använder sig av interfacet NodesToNetwork(som är ett interface) när den skapar nätverket.  
Den skapar hållplatserna genom att skapa objekt av klassen Node som även har en inre klass som heter Path där kopplingarna mellan noderna registreras.  
När nätverket är färdigt och en sökning ska utföras används den abstrakta klassen Dijkstras.  
Denna har en inre klass som heter ShortestPath som efter färdig sökning används för att kunna returnera ett objekt tillbaka till metoden som sen översätter det objektet till text som visas på konsolen.



- Egen kravställning:

* Våra egna krav var att skapa verkliga namn på hållplatserna och även skapa ett symbolnamn för att användaren lättare ska kunna söka. Symbolnamnen är även utformade på ett sätt att oändligt många hållplatser kan finnas i nätverket.
* Vi ville även att användaren ska kunna göra om kopplingarna som en kul grej. Detta görs utan att skapa ett nytt nätverksobjekt och det enda som görs är att det skapas nya kopplingar mellan noderna som finns redan för att undvika nya onödiga objekt.
* Ifall användaren inte skrivit ut nätverket innan kommer det skrivas ut när man söker eftersom användaren måste veta symbolnamnen.
* Som användarvänlighet visar vi också om det är en direktbuss eller om resenären tvingas byta buss och vart dessa byten ska ske. Förutom den totala distansen ser användaren även avstånden mellan byteshållplatserna.

- Kvalitetssäkra arbetet:

För att kvalitetssäkra vårt arbete arbetade vi enligt TDD, frågade projektbeställaren när vi var osäkra på något och testade projektet med nedanstående tester.  
För att ytterligare kvalitetssäkra programmet har vi allteftersom vi byggt metoderna refaktorerat och behövde således inte refaktorera så mycket i efterhand. Vi analyserade koden och justerade då en del detaljer som exempelvis finalvariabler och staticvariabler.

- Testerna:

* Unittester:  
  Det gjorde vi på alla metoder som vi tyckte det behövdes på. Vi utelämnade utskriftsmetoder och userinputsmetoder. Användaren skriver alltid in strings vilket gör metoderna säkra ändå. Många unittester är utkommenterade eftersom metoderna de testar är privata. Vi testade alltså metoderna under utvecklingen men gjorde dom privata när det slutgiltiga projektet skulle lämnas in.
* Användartest:  
  Vi gjorde ett användartest genom att låta en 10-åring testa programmet. Detta lyckades bra och användaren kunde hyfsat väl förstå programmet och även kunna läsa i matrisen vilka hållplatser som var ihopkopplade. En sak som ändå upptäcktes var att det inte stod ”press enter” i menyn. Saker vi själva upptäckte när vi testade var att annan användarvänlig output var på fel plats etc.
* Monkeytest:  
  Vi gjorde monkeytest också och hittade då ett fel om användaren slog mellanslag mellan två ord på input då väljer användaren ett val av menyn.
* Acceptanstest:  
  För att säkerställa projektbeställarens krav gjorde vi ett acceptanstest. Detta gjorde vi genom att noggrant testa programmet utefter kraven(10 noder, randomkoppling etc). Vi ritade även upp det aktuella nätverket med papper och penna och såg till så det var slutet, hade 2-3 noder per nod och att programmet verkligen visade den närmsta vägen.

A close up of text on a black background

Description automatically generated

* Integrationtest & systemtest:  
   Vi hade planer på att köra integrationstest och systemtest men gjorde aldrig detta. Den största anledningen till detta är att programmet skapar nätverket på random och att objektet är singleton och blir då svårt att testa. Dessutom är programmet liten och har inte som många komponenter.

- Ändringar efter refaktorering:

Eftersom det utfördes medan vi skriver metoderna och innan vi pushar så vi hade inte många ändringra som tillkommit efter att vi utförde refaktorering islutet av projektet.  
Den enda ändring som vi gjorde då var att en metod namn ändrades från convertSymbolToInt() till convertSymbolToNode().

- Ändringar i planering och programmet:

Ändringar som skedde under projektets gång var att Dijkstrasklassen blev abstrakt istället för ett interface för vi ville göra de flesta metoderna där privata.  
En annan sak vi lade till var att skapa den inre klassen ShortestPath för att kunna returnera både vikten och hållplatserna mellan start och mål i ett objekt till klassen som anropar sökmetoden.