

Programmering 2: Inlämningsuppgift

Version: 1.3

Uppdaterad: 2022-04-12, se historiken sist i dokumentet för info om ändringar

Författare: Jozef Swiatycki, Patrick Wentzel

Detta är den obligatoriska inlämningsuppgiften på kursen PROG2 VT2022.

Uppgiften är giltig under kursomgången fram till uppsamlingstillfället i augusti 2022 och upphör därefter att gälla och kan komma att ersättas kursomgången 2023.

Uppgiften består av två delar: ett mindre graf-bibliotek¹ (del 1) och ett tillämpningsprogram (del 2).

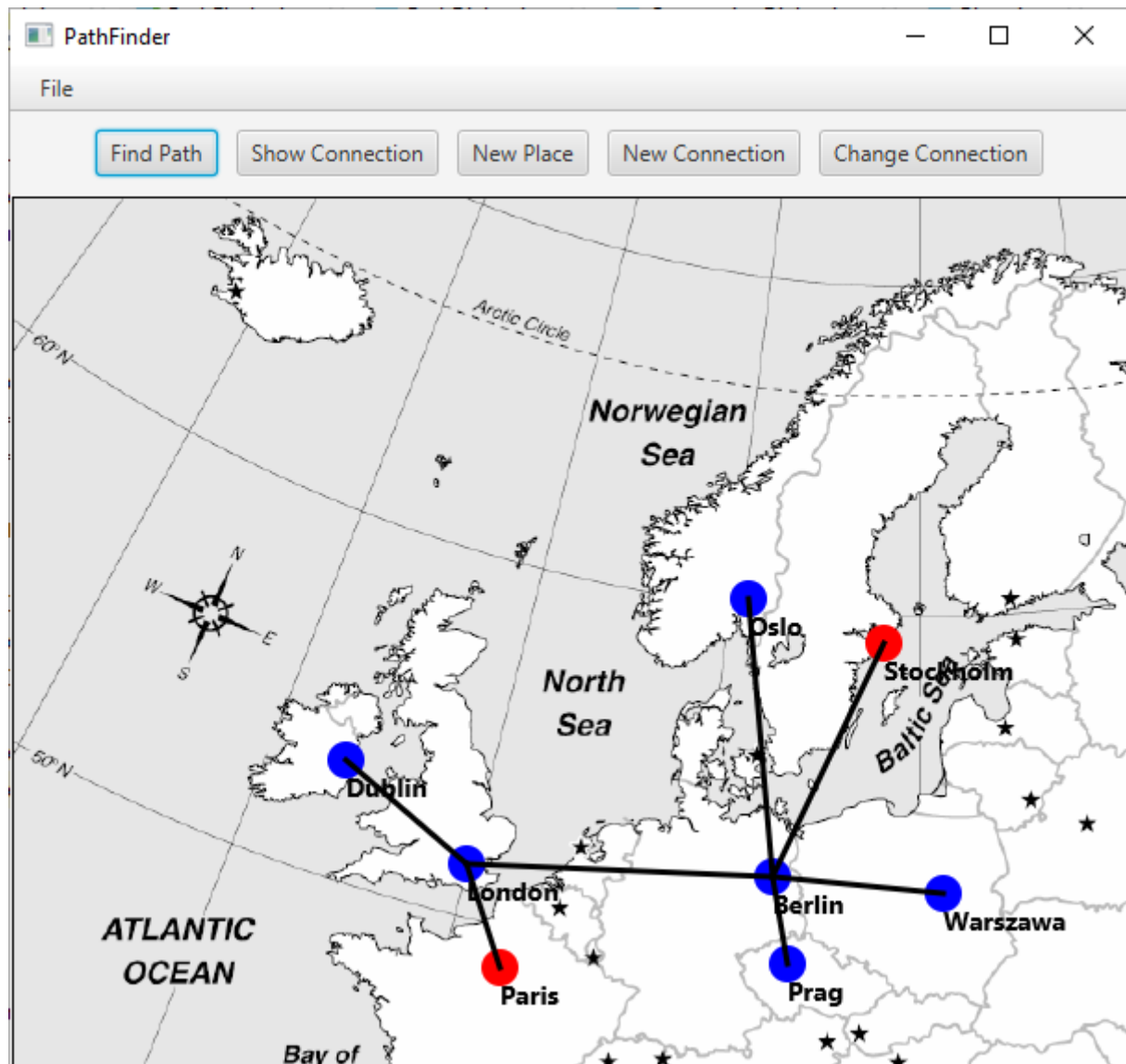
Innehållsförteckning

Inledning	2
Inlämningsdatum (deadlines)	3
Del 1: Datastrukturen ListGraph	4
Klassen ListGraph	4
Klassen Edge	6
Del 2: Ett tillämpningsprogram	7
Menyn File	7
Menyvalet New Map	8
Menyvalet Open	9
Menyvalet Save	10
Menyvalet Save Image	11
Menyvalet Exit	11
Knapparna	12
Knappen New Place	12
Knappen New Connection	14
Knappen Show Connection	15
Knappen Change Connection	16
Knappen Find Path	17
Krav för inlämningen	18
Gemensamt för båda delarna	18
Specifikt för del 1	19
Specifikt för del 2	19
Tips	20
Dokumenthistorik	21

¹ Grafer går igenom på föreläsningar F9-F10.

Inledning

Uppgiften går ut på att skriva ett program som låter användaren öppna en karta (en bakgrundsbild), med musklickningar på kartan definiera platser på kartan samt förbindelser mellan dessa platser och sedan hitta en väg mellan två av användaren valda platser.



Platserna har namn (t.ex. "Kista", "New York", "Andromedagalaxen" o.s.v.). Platserna kan vara förbundna med varandra.

Förbindelserna har namn (t.ex. "Buss 514", "Flyg", "Teleportering" o.s.v.) samt en tid det tar att färdas genom denna förbindelse, dessa (förbindelsens namn och tid) behöver dock inte skrivas ut på kartan.

Användaren kan definiera nya platser på kartan genom att klicka på knappen "Ny plats" och sedan på kartan där platsen ska finnas (se detaljer längre ner). Användaren kan också välja en existerande plats genom att klicka på den, platsen byter då färg.

Användaren kan välja två platser genom att klicka på dem på kartan. När två platser är valda kan användaren definiera en förbindelse mellan dessa platser, visa information om förbindelsen, ändra tiden för förbindelsen eller få en utskrift av vägen mellan dessa två platser

Uppgiften består, som sagt, av två delar: en datastruktur (graf) för att representera nätverket av platser och förbindelser (innehåller inga grafiska operationer) och ett tillämpningsprogram (tar hand om grafiken och om kommunikation med användaren).

Inlämningsdatum (deadlines)

För del 1 är rekommenderat inlämningsdatum den 3/5 och för del 2 är sista inlämningsdatum den 5/6.

Del 1: Datastrukturen `ListGraph`

För att representera nätverket av platser och förbindelser mellan dem behövs en datastruktur graf. Vi kan anta att platserna har ett fåtal förbindelser med andra platser, varför en lösning med kopplingslistor är att föredra framför en lösning med en kopplingsmatris. Det finns inga klasser för graf-representation i Javas standardbibliotek, så i uppgiften ingår att skriva en klass `ListGraph` och en klass `Edge` för representation av enkla² viktade oriktade grafer implementerade med kopplingslistor (*simple weighted undirected graphs implemented by adjacency lists*). Dessa klasser ska skrivas generellt, så att de ska kunna användas av andra, för oss okända tillämpningar.

Klasserna `ListGraph` och `Edge` ska vara generiska, med nod-typen som generisk parameter. Funktionaliteten i `ListGraph` är deklarerad i ett gränssnitt (*interface*) `Graph`.³

Klassen `ListGraph`

Klassen `ListGraph` ska innehålla följande metoder:

- `add` – tar emot en nod och stoppar in den i grafen. Om noden redan finns i grafen blir det ingen förändring.
- `remove`⁴ – tar emot en nod och tar bort den från grafen. Om noden saknas i grafen ska undantaget `NoSuchElementException` från paketet `java.util` genereras. När en nod tas bort ska även dess kanter tas bort, och eftersom grafen är oriktad ska kanterna även tas bort från de andra noderna.
- `connect` – tar två noder, en sträng (namnet på förbindelsen) och ett heltal (förbindelsens vikt) och kopplar ihop dessa noder med kanter med detta namn och denna vikt. Om någon av noderna saknas i grafen ska undantaget `NoSuchElementException` från paketet `java.util` genereras. Om vikten är negativ ska undantaget `IllegalArgumentException` genereras. Om en kant redan finns mellan dessa två noder ska undantaget `IllegalStateException` genereras (det ska finnas högst en förbindelse mellan två noder).

Observera att grafen ska vara oriktad, d.v.s. en förbindelse representeras av två kanter: kanter riktade mot den andra noden måste stoppas in hos de båda noderna. I en oriktad graf förekommer ju alltid kanter i par: från nod 1 till nod 2 och tvärtom.

- `disconnect` – tar två noder och tar bort kanten som kopplar ihop dessa noder. Om någon av noderna saknas i grafen ska undantaget

² Påminnelse: enkla grafer är grafer utan parallella kanter eller kanter till start-noden själv.

³ Det kan vara klokt att vänta med att låta `ListGraph` implementera `Graph` tills hela gränssnittet är implementerat, så att man kan skriva och testa ut `ListGraph`-metoderna var för sig.

⁴ `remove` och `disconnect` används enbart i del 1 och utnyttjas inte i del 2.

`NoSuchElementException` från paketet `java.util` genereras. Om det inte finns en kant mellan dessa två noder ska undantaget `IllegalStateException` genereras. (Här kan säkert metoden `getEdgeBetween` vara till nytta.)

Observera att eftersom grafen är oriktad, d.v.s. en förbindelse representeras av två kanter så måste kanten tas bort från båda noderna.

- `setConnectionWeight` – tar två noder och ett heltal (förbindelsens nya vikt) och sätter denna vikt som den nya vikten hos förbindelsen mellan dessa två noder. Om någon av noderna saknas i grafen eller ingen kant finns mellan dessa två noder ska undantaget `NoSuchElementException` från paketet `java.util` genereras. Om vikten är negativ ska undantaget `IllegalArgumentException` genereras.
- `getNodes` – returnerar en kopia av mängden av alla noder.
- `getEdgesFrom` – tar en nod och returnerar en kopia av samlingen av alla kanter som leder från denna nod. Om noden saknas i grafen ska undantaget `NoSuchElementException` genereras.
- `getEdgeBetween` – tar två noder och returnerar kanten mellan dessa noder. Om någon av noderna saknas i grafen ska undantaget `NoSuchElementException` genereras. Om det inte finns någon kant mellan noderna returneras `null`.
- `toString` – returnerar en lång sträng med strängar tagna från nodernas `toString`-metoder och kanternas `toString`-metoder, gärna med radbrytningar så att man får information om en nod per rad för förbättrad läsbarhet.
- `pathExists` – tar två noder och returnerar `true` om det finns en väg genom grafen från den ena noden till den andra (eventuellt över många andra noder), annars returneras `false`. Om någon av noderna inte finns i grafen returneras också `false`. Använder sig av en hjälpmetod för djupet-först-sökning genom en graf.
- `getPath` – tar två noder och returnerar en lista (`java.util.List`) med kanter som representerar en väg mellan dessa noder genom grafen, eller `null` om det inte finns någon väg mellan dessa två noder. I den enklaste varianten räcker det alltså att metoden returnerar någon väg mellan de två noderna, men frivilligt kan man göra en lösning där returnerar den kortaste vägen (i antalet kanter som måste passeras) eller den snabbaste vägen (med hänsyn tagen till kanternas vikter).

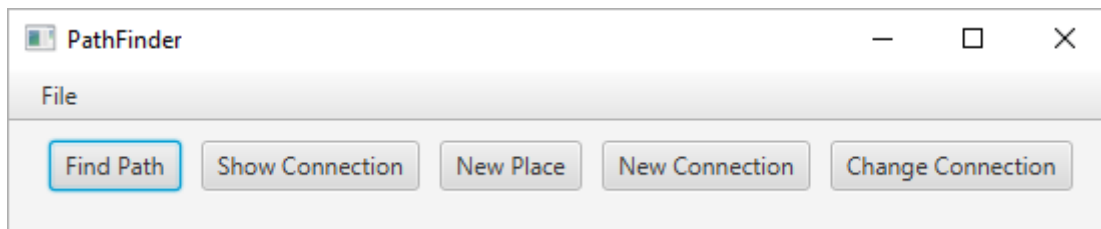
Klassen Edge

Klassen `Edge` ska representera en kant, alltså den ena riktningen av en förbindelse mellan två noder. Värden för alla instansvariabler i denna klass ska skickas som argument till konstruktorn. Klassen ska innehålla följande metoder:

- `getDestination` – returnerar den nod som kanten pekar till.
- `getWeight` – returnerar kantens vikt
- `setWeight` – tar ett heltal och sätter kantens vikt till detta heltal. Om vikten är negativ ska undantaget `IllegalArgumentException` genereras.
- `getName` – returnerar kantens namn
- `toString` – returnerar en sträng med information om kanten.

Del 2: Ett tillämpningsprogram⁵

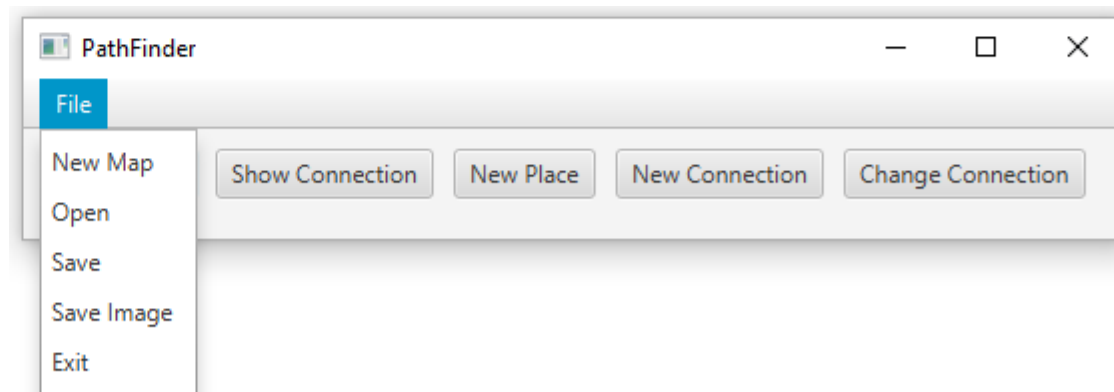
När man först startar programmet ska ett fönster öppnas med följande utseende:



Fönstret har en meny "File" och fem knappar för de olika operationerna användaren kan begära. "File"-menyn ska innehålla menyvalen "New Map", "Open", "Save", "Save Image" och "Exit".

Menyn File

Nedan visas programfönstret med "File"-menyn nedfälld:

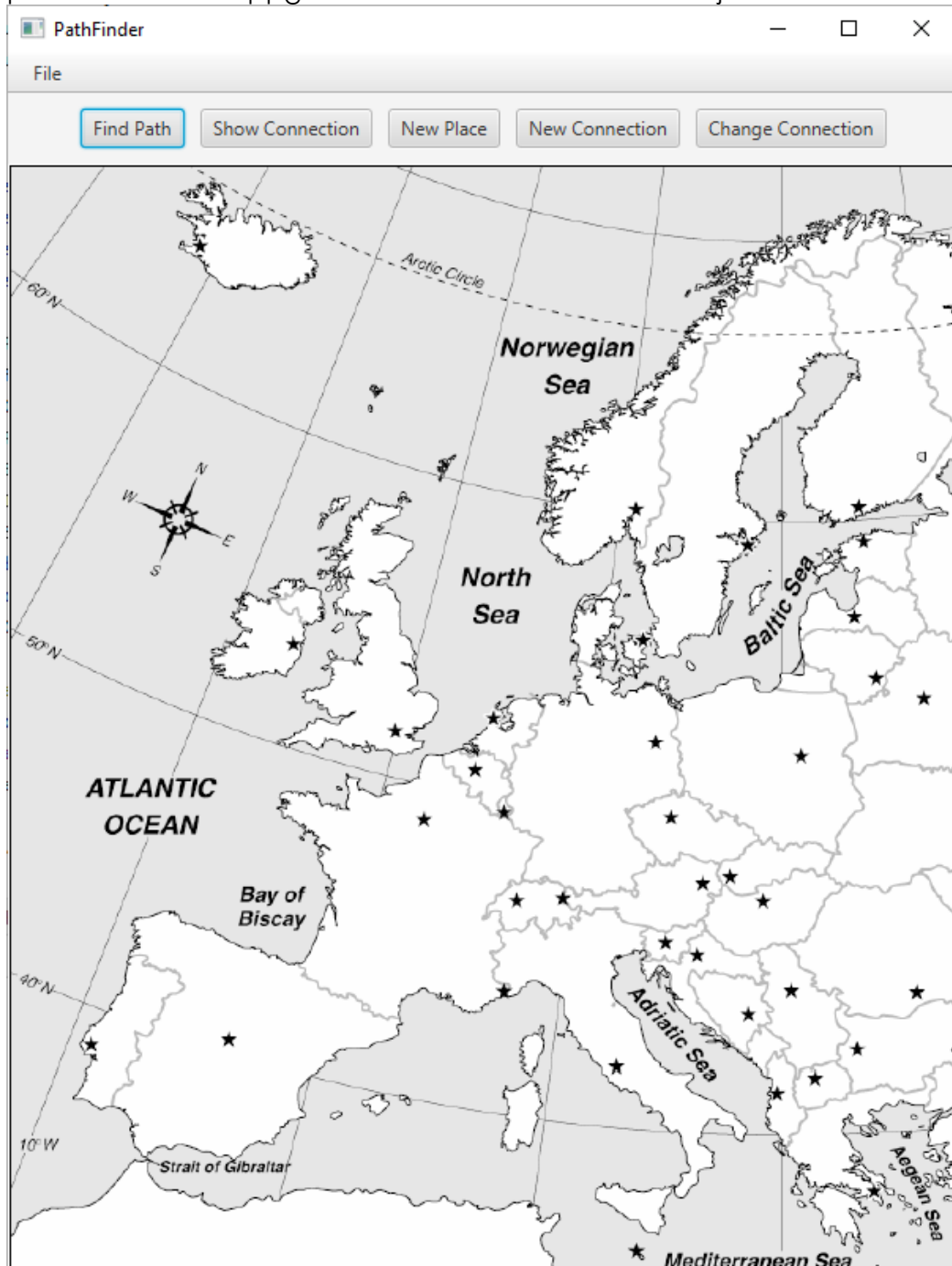


Egentligen borde menyvalen "New Map", "Open" och "Save" öppna en fildialog där användaren kan välja en fil att arbeta med, men det leder till svårigheter vid rättning av inlämningsuppgiften. Av rättningstekniska skäl ska alltså programmet inte fråga användaren om filnamnet utan jobba på hårdkodade filnamn.

⁵ I den här delen av inlämningsuppgiften används komponenter och tekniker som går igenom på föreläsningarna F11-F16.

Menyvalet New Map

"**New Map**" innebär att användaren vill börja arbeta med en tom karta. Själva kartan är bara en bakgrundsbild som ska visas i fönstrets centrala area. Bilden ska läsas in från URL "file:europa.gif" i projekt-mappen. Bildfilen finns i iLearn på samma plats som denna uppgiftstext. Fönstret ska se ut som följer:



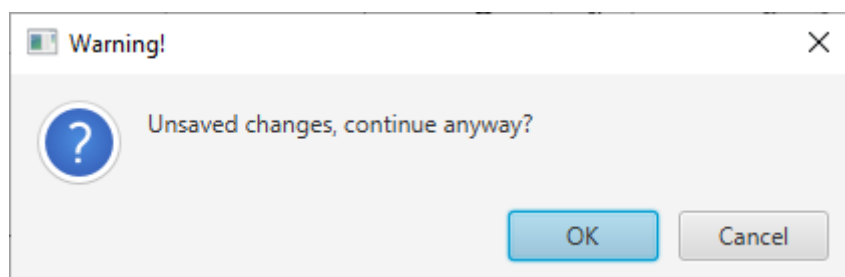
Kartan är som sagt bara en bakgrundsbild. Fönsterstorleken bör anpassas till bilden så att hela kartan visas. Obs att kartan inte bör förändras om användaren förstorar eller förminskar fönstret (eftersom platser kommer att definieras på fasta koordinater på kartan skulle deras position på kartan förändras om man tillät att kartan skalades om).

Menyvalet Open

"Open" ska öppna filen *"europa.graph"* som tidigare ska ha skapats genom menyvalet **"Save"**.

Om det inte finns någon sådan fil ska ett felmeddelande ges. Annars laddas kartan och objekten (platserna och förbindelserna) in och visas i fönstret. Filen ska ligga på toppnivå i projekt-mappen, den ska vara en textfil enligt det format som beskrivs under menyvalet **"Save"**. Den ska innehålla filnamnet på bildfilen med kartan, information om alla noder (namn och koordinater) och information om alla förbindelser (från-nod, till-nod, namnet och vikten). Filtypen för denna fil ska vara *".graph"*. Med hjälp av denna information ska man kunna återskapa kartan med dess platser och förbindelser mellan platserna.

Obs dock att om användaren tidigare hämtat en karta eller öppnat en fil och definierat nya platser eller förbindelser, eller ändrat tiden för en förbindelse – kort sagt gjort några förändringar och inte sparat så ska följande dialog visas:



Vid svaret **"OK"** förkastas de osparade ändringarna och det nya **"dokumentet"** öppnas, vid **"Cancel"** avbryts operationen. Ovanstående gäller även vid **"New Map"**, vid **"Exit"** och när användaren klickar i fönstrets stängningsruta. Programmet bör alltså hålla reda på om det finns osparade ändringar.

Menyvalet Save

"Save" ska spara data om objekten på en textfil med namnet "europa.graph" på följande format: först ska det vara en rad med URL till bildfilen med kartan, därefter ska det vara en lång rad med semikolonseparerade uppgifter om noder, och därefter flera rader med uppgifter om förbindelserna, en förbindelse per rad. Uppgifter om noder ska omfatta nodens namn, nodens x-koordinat och nodens y-koordinat, separerade med semikolon. Raderna med uppgifter om förbindelserna ska för varje förbindelse innehålla namnet på från-noden, namnet på till-noden, namnet på förbindelsen och förbindelsens vikt, separerade med semikolon.

Ett exempel på en sådan fil ges här (obs att det inte finns några mellanslag efter semikolon):

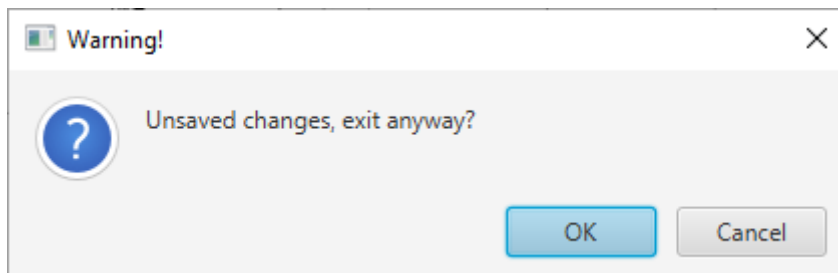
```
file:europa.gif
Stockholm;469.0;242.0;Oslo;398.0;219.0;Warszawa;503.0;377.0;Berlin;410.0;367.0
Stockholm;Oslo;Train;3
Stockholm;Warszawa;Airplane;2
Oslo;Stockholm;Train;3
Warszawa;Stockholm;Airplane;2
Warszawa;Berlin;Train;6
Berlin;Warszawa;Train;6
```

Menyvalet Save Image

"**Save Image**"⁶ ska spara en ögonblicksbild av fönstrets innehåll på en fil med namnet "capture.png" på PNG-formatet. Filen ska läggas på toppnivå i projektmappen.

Menyvalet Exit

"**Exit**" ska avsluta programmet. Programmet ska dock hålla reda på om det finns osparade ändringar och i så fall varna användaren:



Se beskrivningen för "Open" för hur programmet ska bete sig.

Programmet ska också kunna avslutas genom att klicka i fönstrets stängningsruta, med samma beteende.

⁶ Koden för hur detta görs visas på en av föreläsningarna F11-F16.

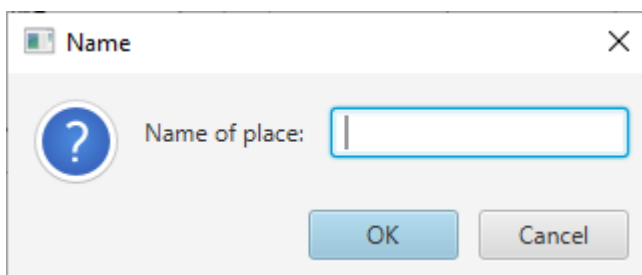
Knapparna

Knapparna ska bara kunna användas efter att en karta har laddats in i programmet.

Knappen New Place

"New Place" ska som namnet antyder användas för att lägga till platser på kartan. Operationen ska dock göras i några steg: först klickar användaren på knappen. Detta gör att muspekaren ändras till ett + (crosshair) i väntan på att användaren klickar på den position på kartan där hen vill ha sin plats. Samtidigt ska knappen göras inaktiv ("disableas").

Efter att användaren klickat på kartan ska en dialogruta dyka upp:

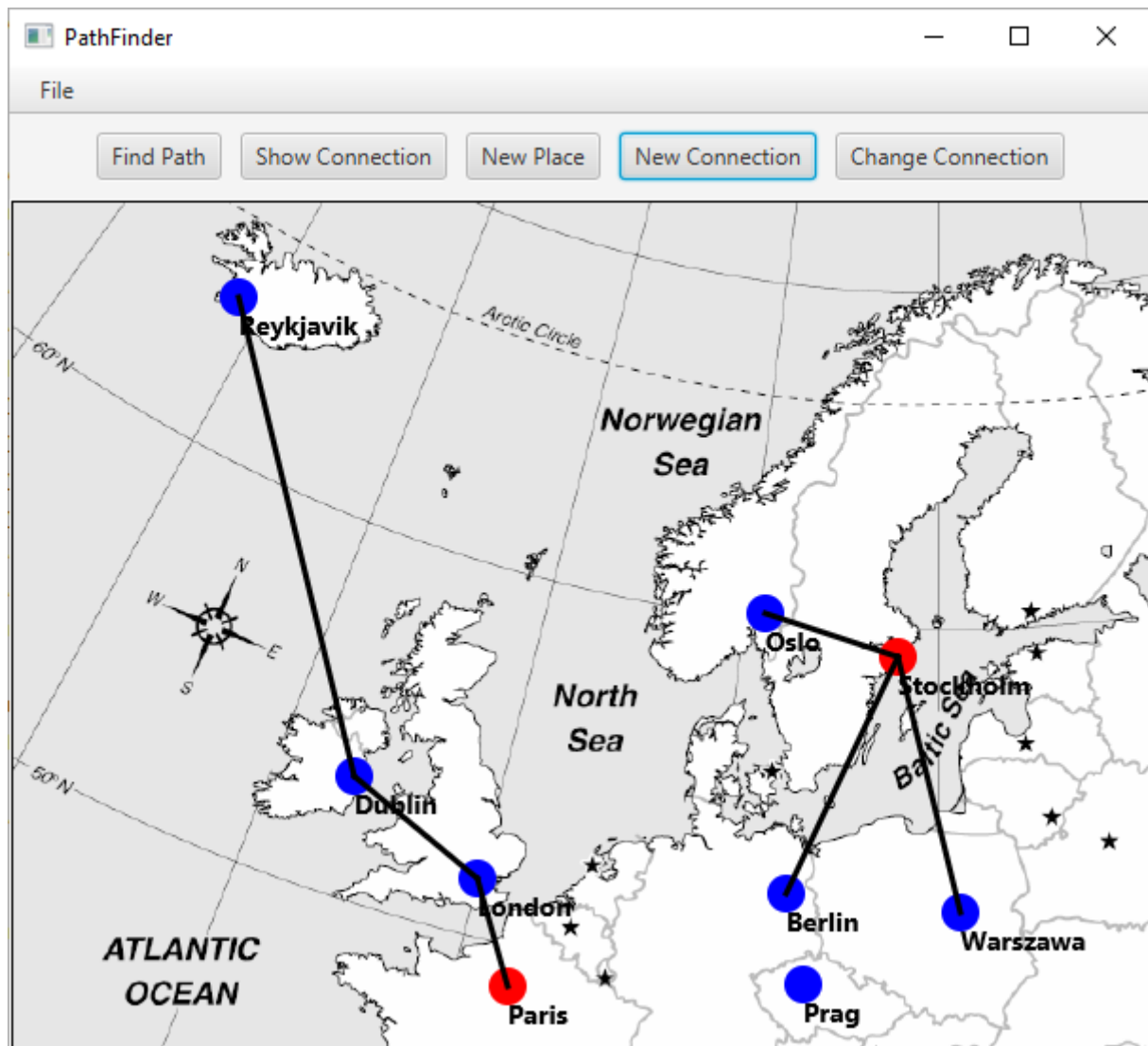


När användaren klickat på "OK" ska den nyskapade platsen dyka upp på kartan. Muspekaren ska återgå till att vara en vanlig pil och knappen ska aktiveras igen ("enablas").

Användaren ska kunna markera platser genom att klicka på dem. Platserna ska då byta färg från blå till röd. Om man klickar på platsen en gång till så avmarkeras den (och byter färg till blå igen). Det ska aldrig gå att markera fler än två platser åt gången, försöker man markera en tredje plats så ska ingenting hända.

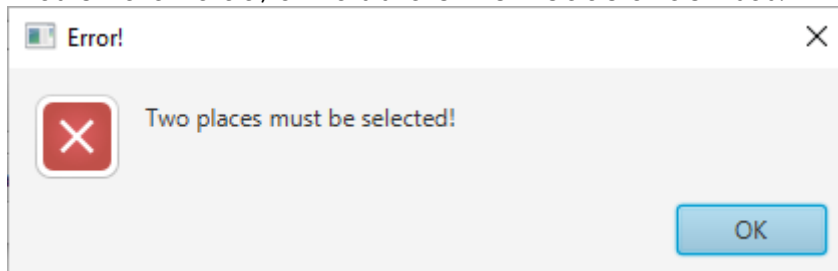
Platsen som markerats först blir från-platsen för en förbindelse, den andra markerade platsen blir till-platsen i förbindelsen. I exemplet nedan är Stockholm och Paris de valda platserna och man kan definiera en förbindelse mellan dem genom att klicka på knappen "New Connection".

Så här skulle fönstret kunna se ut efter att några platser och förbindelser hade lagts till:

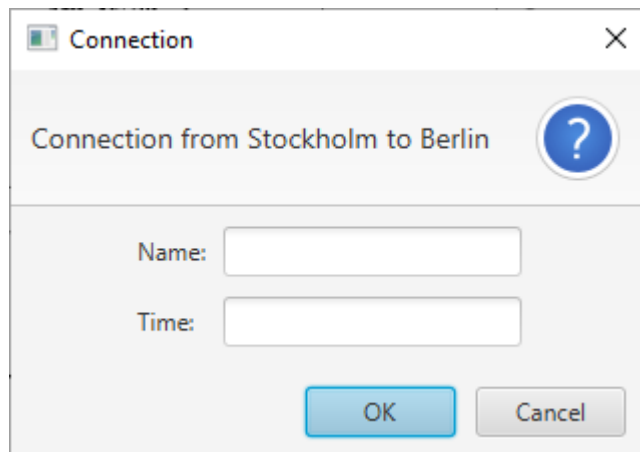


Knappen New Connection

"**New Connection**" ska användas för att skapa förbindelser mellan platser. Två platser måste vara valda, annars ska ett felmeddelande visas:



Om det redan finns en förbindelse mellan de valda platserna ska också ett lämpligt felmeddelande visas (det kan bara finnas en förbindelse mellan två platser).



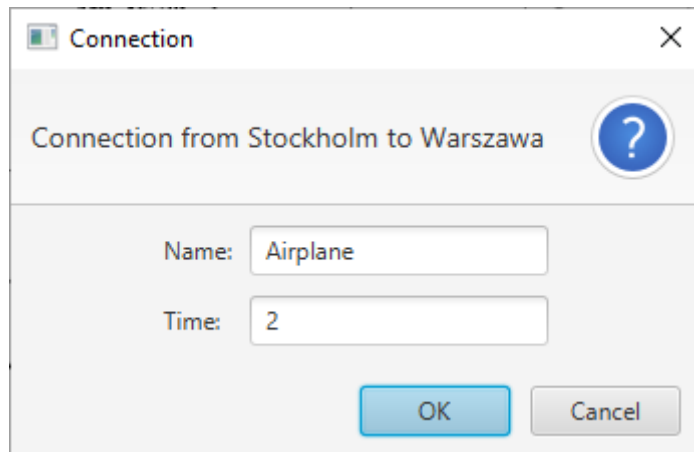
Annars ska ett dialogfönster visas, där användaren kan mata in namnet på förbindelsen och hur lång tid (heltal, t.ex. antal timmar) det tar att ta sig igenom förbindelsen. Av dialogen ska framgå mellan vilka platser förbindelsen skapas:

Om användaren trycker på "OK" ska inmatningen kontrolleras. Namnfältet får inte vara tomt och tidfältet måste bestå av siffror. Om inmatningen uppfyller dessa villkor ska förbindelsen skapas. Om inmatningen inte uppfyller villkoren ska ett felmeddelande ges och operationen ska avbrytas.

Om användaren trycker på "Cancel" ska operationen avbrytas.

Knappen Show Connection

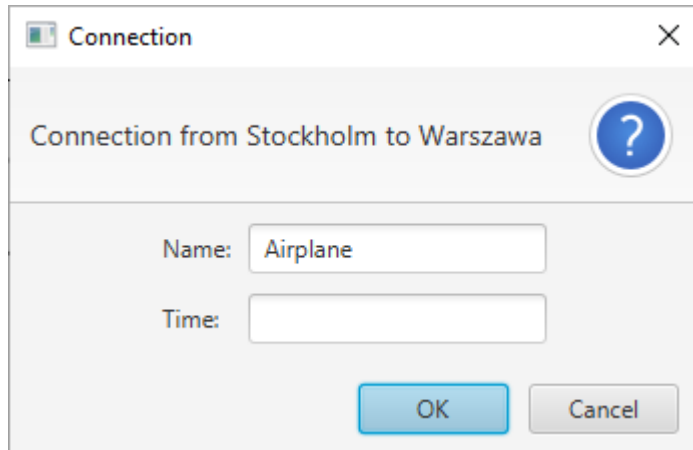
"Show Connection" visar uppgifter om förbindelsen mellan de två valda platserna eller ett felmeddelande (se ovan) om två platser inte är valda, eller ett felmeddelande om det inte finns någon förbindelse mellan platserna. Fönstret med uppgifter om förbindelsen kan t.ex. se ut så här (obs. att både textfälten är icke-editerbara):



The screenshot shows a Windows-style dialog box titled "Connection" with a close button (X) in the top right corner. The main content area has a light gray background. At the top of this area, the text "Connection from Stockholm to Warszawa" is displayed in a blue font. To the right of this text is a blue circular icon containing a white question mark. Below this header, there are two text labels, "Name:" and "Time:", each followed by a white text input field. The "Name:" field contains the text "Airplane" and the "Time:" field contains the number "2". At the bottom of the dialog box, there are two buttons: a blue "OK" button and a gray "Cancel" button.

Knappen Change Connection

"Change Connection" tillåter ändring av tiden för en förbindelse. Visar först namnet på förbindelsen mellan de två valda platserna eller ett felmeddelande (se ovan) om två platser inte är valda, eller ett felmeddelande om det inte finns någon förbindelse mellan platserna. Fönstret med uppgifter om förbindelsen kan t.ex. se ut så här (obs. att textfältet för namn är icke-editerbart):

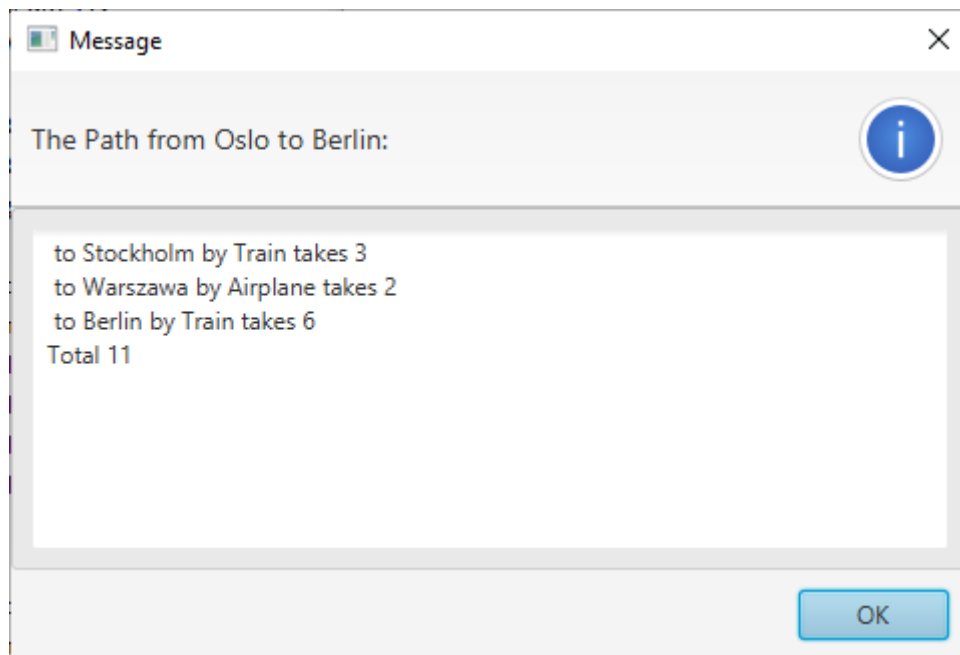


The screenshot shows a Windows-style dialog box titled "Connection". The title bar includes a standard close button (X). Below the title bar, the text "Connection from Stockholm to Warszawa" is displayed, followed by a blue circular icon containing a white question mark. The main area of the dialog contains two labels: "Name:" and "Time:". The "Name:" label is followed by a text field containing the word "Airplane". The "Time:" label is followed by an empty text field. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

Tänk på att grafen är oriktad: ändringar i förbindelsen åt ena hållet måste återspeglas i förbindelsen åt andra hållet.

Knappen Find Path

"Find Path" används för att söka igenom grafen efter en väg mellan de två valda platserna. Detta sker genom att programmet använder sig av den relevanta metoden i graf-klassen och skriver ut resultatet i en dialogruta. Dialogrutan ska innehålla all relevant information om resan, alltså var man börjar och slutar, vilka platser och förbindelser som passeras, hur lång tid varje delsträcka tar samt hur lång tid resan tar totalt:



Om det inte finns någon väg mellan de två platserna ska det istället dyka upp en dialogruta som meddelar detta.

Obs att det inte är bestämt i uppgiften vilken väg som visas av "Find Path": det kan vara vilken väg som helst, den kortaste vägen eller den snabbaste vägen. Det beror på hur du har löst metoden `ListGraph.getPath()` i första delen av inlämningsuppgiften och är alltså valfritt.

Krav för inlämningen

För att inlämningarna ska kunna testas, och för del 1 även rättas, automatiskt finns det en del krav som måste vara uppfyllda.

Gemensamt för båda delarna

Inlämning sker precis som på PROG 1 genom VPL i olika inlämningslådor för de olika uppgifterna.

För att kunna lämna in behöver man ha valt en grupp, och efter att man har lämnat in kan man inte byta grupp (för den inlämningen).

Det som ska lämnas in är enbart Java-källkodsfiler. Inget annat. I vissa uppgifter ställs krav på specifika namn på klasser och filer.

Vid inlämning kommer koden att utvärderas på olika sätt med bl. a. CheckStyle och sedan kommer funktionen testas med enhetstester (JUnit).

Överst i samtliga inlämnade filer ska det finnas en kommentar med namn och användarnamn för samtliga gruppmedlemmar. Exempelvis:

```
// PROG2 VT2021, Inlämningsuppgift, del 1
// Grupp 004
// Dewey Duck dedu1111
// Huey Duck hudu1234
```

Notera att all kod som lämnas in måste vara er egen. Allt fusk rapporteras.

Det ställs krav på kodutseende: koden ska vara korrekt indenterad, namn på klasser, variabler och metoder ska följa Javas namngivningskonventioner, variabler i de klasser som är tänkta som biblioteksklasser ska vara private-deklarerade och metoderna i klasserna ska ha korrekta skyddsnivåer (public, protected eller private).

Varje klass och interface ska finnas i en egen fil (utom om det finns ett motiverat behov av anonyma inre klasser).

Specifikt för del 1

För del ett gäller inte särskilt utöver det som nämns ovan, men det är *extremt* viktigt att man följer det givna interfacet Graph.java exakt.

Specifikt för del 2

För del två gäller att man dels använder just de texter som nämns ovan i sina menyval och för knapparna. Det ska vara just som det står i beskrivningarna.

Grafen behöver innehålla samma objekt som är klickbara på kartan⁷.

Samtliga noder (JavaFX-komponenter som ärver från Node), dvs. knappar samt den komponent som utgör ytan (förmodligen en Pane), ska ha ett ID satt med metoden setId enligt följande:

Komponent	ID
Menyn (en menuBar)	menu
File-menyn	menuFile
Menyvalet New Map	menuNewMap
Menyvalet Open	menuOpenFile
Menyvalet Save	menuSaveFile
Menyvalet Save Image	menuSaveImage
Menyvalet Exit	menuExit
Knappen Find Path	btnFindPath
Knappen Show Connection	btnShowConnection
Knappen New Place	btnNewPlace
Knappen Change Connection	btnChangeConnection
Knappen New Connection	btnNewConnection
Den komponent som platser läggs till på (ritytan)	outputArea
Platser som läggs till på kartan	<i>namnet på platsen</i>

⁷ Detta beror på att testprogrammet behöver kunna klicka på två noder på kartan och sedan använda de markerade objekten för att se om det finns en båge eller väg mellan platserna genom att konsultera grafen.

Tips

I samband med testningen har vi uppmärksammat en del ofta förekommande problem.

Problem: testet avbryts direkt med meddelande om att någon komponents id inte kunde hittas (t.ex. outputArea).

En tänkbar orsak är att komponenten inte är tillagd i scen-grafen från början utan läggs till först i ett senare steg, t.ex. när ny karta laddas in. Lösningen är att lägga till alla delar redan i start-metoden.

Problem: en plats blir inte markerad eller avmarkerad som den borde.

Det här kan t.ex. orsakas av att någon annan komponent ligger ovanpå platsen och "stjäl" klicket. Det skulle kunna vara en plats som borde ha varit borttagen i samband med att en ny karta eller ny graf laddats in, eller en text/label eller linje som ligger ovanpå. Allt som läggs till på kartan som inte ska vara klickbart bör inaktiveras (`setDisable(true)`).

En annan anledning kan vara att det som håller reda på vad som är markerat kanske inte fungerar helt, eller inte nollställs när ny karta eller ny graf laddas in.

Problem: filer verkar inte kunna laddas in (antingen karta eller graf)

Detta kan orsakas av felaktig sökväg. Klassen `Image` vill ha en URL: `file:europa.gif` men `File` och `FileReader` vill ha ett filnamn: `europa.graph`.

Problem: när ny graf-fil laddas in blir det fel med kartan.

I samband med att graf-filen `europa.graph` laddas in ska bilden med kartan bytas ut till den som anges på första raden i filen. Filnamnet `file:europa.gif` ska inte vara hårdkodat i inläsningen.

Dokumenthistorik

Version	Datum	Ändringar
1.0	2021-03-23	Första version.
1.1	2021-03-23	Rättat exempel för kommandot Save
1.2	2021-05-17	Lagt till sektion med tips
1.3	2022-04-13	Uppdaterat för VT22.