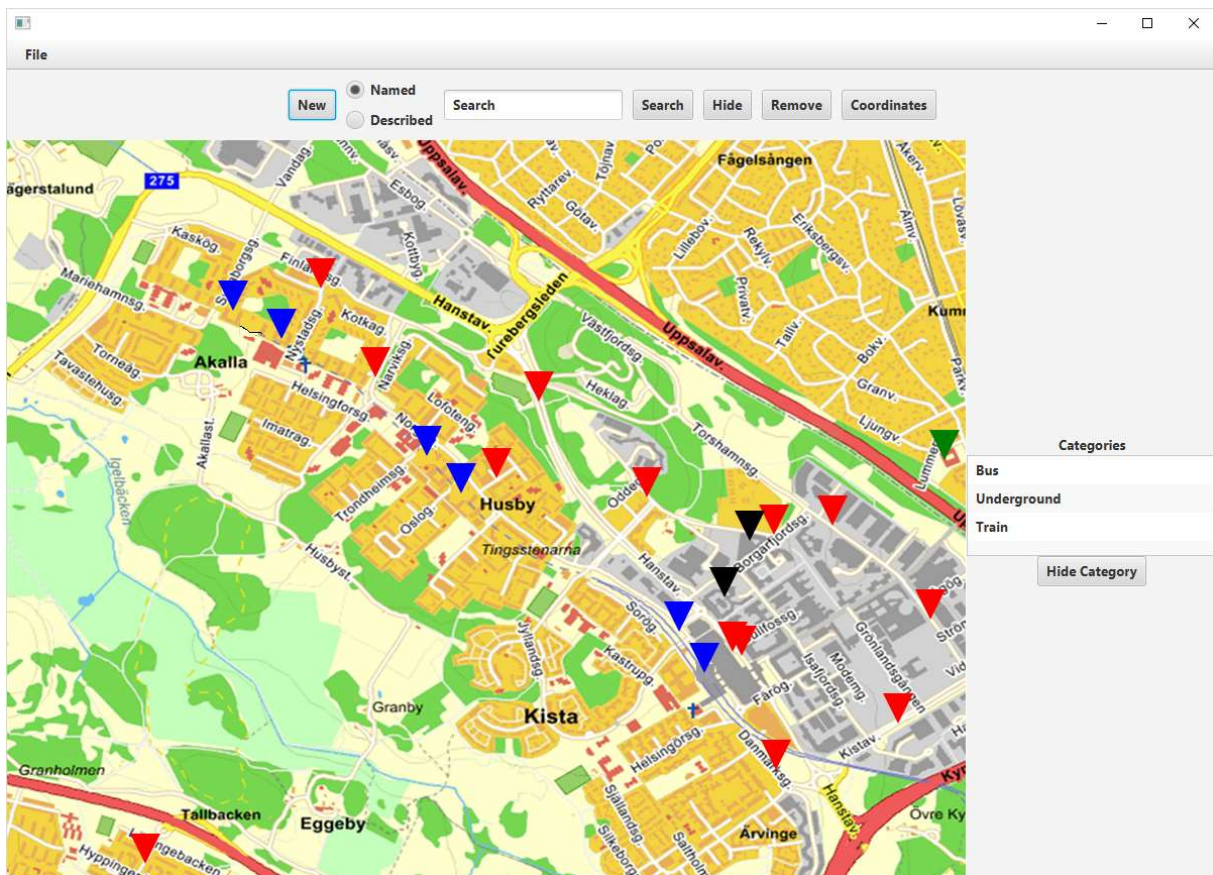


Inlämningsuppgift 2

Denna uppgift är den andra (av två) obligatoriska inlämningsuppgiften på delkursen PROG2 Programmering 2 VT2019. Uppgiften ska lösas enskilt eller i grupper om högst tre personer. Inlämningsinstruktionen finns sist i det här dokumentet.

Uppgiften går ut på att skriva ett program som låter användaren öppna en karta (en bakgrundsbild) och med musklickningar definiera platser på kartan. Platserna har namn, kan ha en beskrivande text och kan tillhöra någon av de tre kategorierna Bus, Underground eller Train¹, men de kan också vara kategorilösa. Uppgiften är inspirerad av användargränssnittet på kartor.eniro.se, men är givetvis mycket förenklad och implementeras helt annorlunda: kartan är bara en bild, det finns ingen zoomning, man kan inte flytta på kart (den är inte större än vad som visas) osv.an



På ovanstående bild är det de olikfärgade trianglarna som visar platser som användaren har definierat, tanken är att triangelns spets pekar på punkten (pixeln) där platsen är. Färgen på triangeln bestäms av vilken kategori (Bus, Underground, Train eller ingen) som platsen tillhör, vilket i sin tur bestäms av vilken kategori som var vald i listan till höger när platsen skapades.

Platserna kan vara av två typer: NamedPlace (har bara ett namn) eller DescribedPlace (har ett namn och en beskrivning). I en tänkt framtida utökning ska man kunna ha fler typer av platser, t.ex. platser med en tidtabell, platser med en bild osv.

¹ På förekommen anledning: du får inte ändra namn på kategorierna, för då kommer inte våra tester att fungera

Användare ska kunna utföra några operationer på platserna: markera platser, gömma markerade platser, ta bort markerade platser, gömma eller visa alla platser av en viss kategori, söka efter platser med ett visst namn och kolla vad som finns på angivna koordinater.

Meningen med uppgiften är att öva dels på GUI med inslag av egenritad grafik, dels på datastrukturer. Platser ska givetvis samlas i några datastrukturer. I en primitiv lösning skulle datastrukturen kunna vara en `ArrayList` som man söker igenom sekventiellt varje gång man behöver göra något med platserna. Detta är dock en ineffektiv lösning om antalet platser är stort.

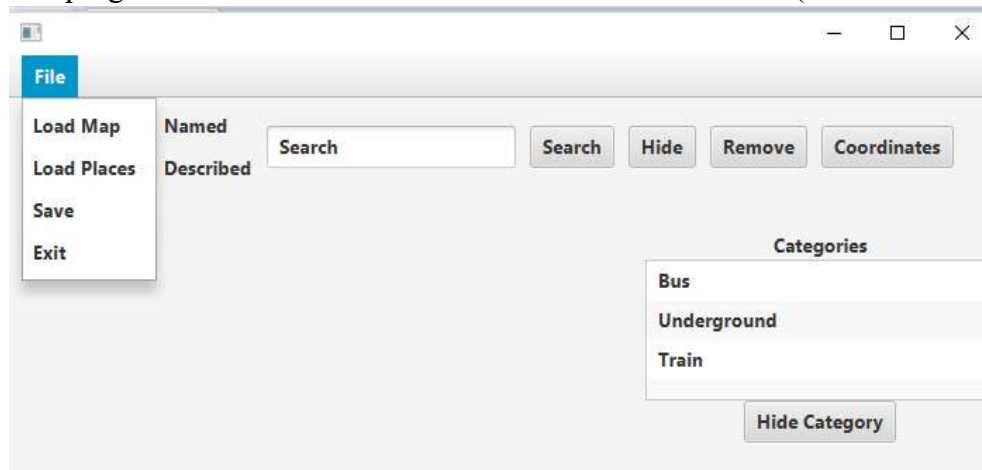
Det är **inte tillåtet** med en sådan lösning på denna uppgift.

För varje sökning eller annan operation skall du tänka igenom om operationen skulle underlättas med någon speciell datastruktur och i så fall använda denna datastruktur. T.ex. vid hantering av kategorier bör det finnas en datastruktur som samlar alla platser för varje kategori, vid sökning på namn bör det finnas en datastruktur som samlar alla platser med samma namn osv. Du bör utforma lösningen som om antalet platser kunde vara mycket stort.

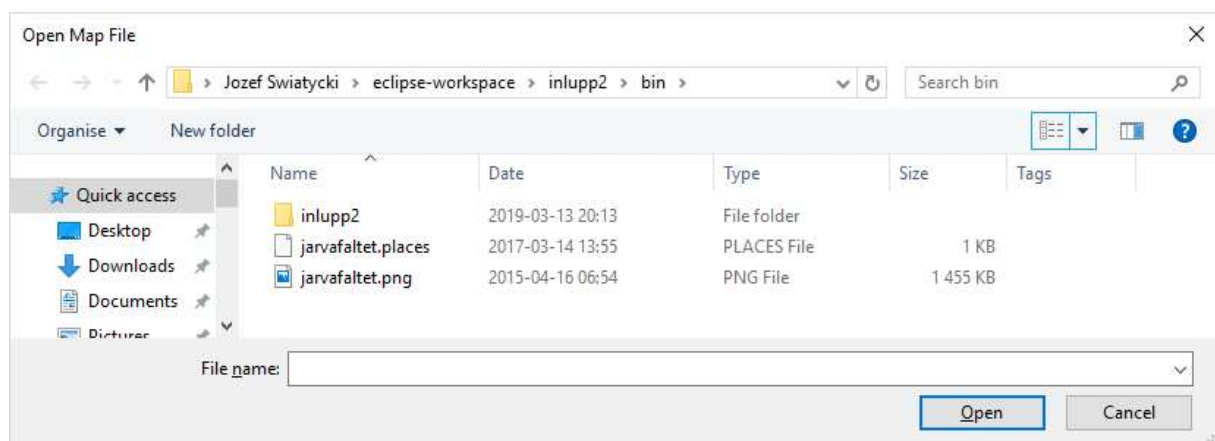
Kartan

Kartan är en vanlig bildfil.

När programmet startar visas ett fönster som kan se ut så här (visas här med menyn File nedfällt):



Om man väljer New Map så visas en fildialog där användaren kan välja en bild (förhoppningsvis föreställande en karta):



Bilden ska laddas in och visas i fönstret. Man ska kunna förstora fönstret genom att dra ut det så att man kan se en större del av kartan – kartan ska inte skalas om då.

Obs! Tillsammans med denna uppgiftstext finns filen `jarvafaltet.png` som innehåller en bild med en karta över Norra Järva. Det finns också en fil `jarvafaltet.places` med sparade platser (se File-menyn nedan) som gäller just denna karta.

Position

Varje plats har ett namn som anges då platsen skapas, en position och en färg (som bestäms av kategorin). Positionen anger helt enkelt platsens pixelkoordinater och ska representeras med en egenskriven klass med namnet `Position`. Objekt av denna klass omsluter alltså bara två koordinater, `x` och `y`, representerade som två doubles.

En `Position` ska kunna användas för att identifiera platser när man vill veta vad som ligger på en viss position. `Position`-klassen ska därför förberedas så att dess objekt ska fungera väl som nycklar i hashtabeller.²

Obs den lilla komplikationen att platsens grafiska area räknas från dess övre vänstra hörn, medan platsens position enligt denna tillämpning är triangelns nedre spets. Det kräver lite omräkning av koordinater.

Platser

Platser är alltså tänkta att vara av olika typer, innehållande olika typer av information. En objektorienterad lösning på detta är att göra en klasshierarki, där de olika platstyperna är subklasser till en superklass `Place`. I uppgiften ingår att det ska finnas två typer av platser: `NamedPlace` som (förutom koordinater, kategori osv) bara har ett namn och en `DescribedPlace` som (förutom koordinater, kategori osv) har både ett namn och en beskrivande text.

En plats skapas genom att användaren väljer kategori i listan till höger, väljer platstypen med hjälp av radioknapparna vid `New`-knappen och trycker på knappen `New`. Då ska markören över kartan ändras till ett kors (för att markera att nästa klick på kartan skapar en plats) och en klick på kartan skapar en plats på den klickade positionen. Obs att det är tänkt att den nedre triangelspetsen visar var platsen finns, så det behövs en viss justering av koordinater för platsen. Om ingen kategori är markerad när en plats skapas blir platsen kategorilös och dess färg blir svart.

Om det redan finns en plats på den klickade punkten ska ett felmeddelande ges – det är endast tillåtet med en plats per position. Se beskrivningen av operationen `Coordinates` nedan för lite mer information om detta.

Efter att platsen är skapad ska inte kartan vara mottaglig för klickning förrän användaren trycker på knappen `New` igen.

När en `NamedPlace` skapas ska användaren ange platsens namn. När en `DescribedPlace` skapas ska användaren dels ange namnet, dels en beskrivning som (av tekniska skäl) består av bara en rad.

² Det finns en liknande klass `Point` i Javas bibliotek, men för övnings skull är det alltså obligatoriskt att skriva en egen sådan klass istället.

En plats ska kunna vara synlig eller gömd (man kan gömma platser man är ointresserad av för att göra kartan mer överskådlig) och den ska kunna vara markerad eller ej. Dessutom ska användaren kunna högerklicka på en plats och då få se dess information i en dialogruta.

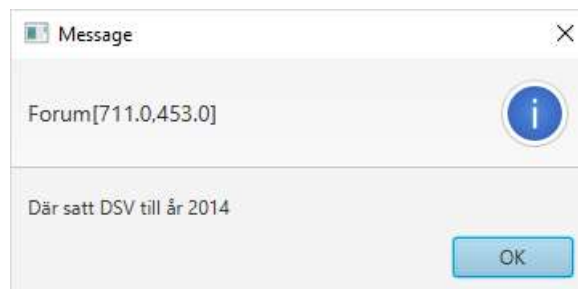
Hantering av om platsen är synlig eller inte kan implementeras med anrop till grafiska komponenters `setVisible`-metod (`Place`-klassen bör givetvis ärvas från någon subklass till `Node`). Om platsen visas markerad eller inte ska implementeras i `Place`-klassen.

Användaren ska kunna markera/avmarkera en plats genom att klicka på den med vänster musknapp³. Flera platser ska kunna vara markerade samtidigt. Hur en markerad plats visas kan du bestämma själv, exempelvis kan markerade platser visas i färgen gul och med svarta kanter.

Användaren ska kunna klicka på en plats med höger musknapp och då få se information om platsen. Informationen visas i en dialogruta. Vilken information som visas beror på platstypen: för en `NamedPlace` visas namnet och koordinaterna⁴:



För en `DescribedPlace` visas namnet med koordinater, samt beskrivningen:



Om denna visning ordnas med hjälp av `Alert`-dialogrutor så är det lättast att informationen visas bara för en plats åt gången – detta är en tillåten begränsning i uppgiften.

Programmet måste kunna hålla reda på vilka platser som är markerade – operationerna `Hide` och `Remove` arbetar på de markerade platserna.

Search

Operationen `Search` letar upp platser med det namn som matats in i sökfältet. Den börjar med att avmarkera ev. platser som är markerade före sökningen, därefter hämtar den söksträngen från sökfältet, och gör alla platser som har detta namn synliga och markerade. Resultatet av sökningen presenteras alltså genom att platser med detta namn blir markerade (och synliga om de var gömda förut).

³ I en `handle`-metod för `MouseEvent` kan man kolla vilken knapp som trycktes på så här:

`if (event.getButton() == MouseButton.PRIMARY)` – det var vänster musknapp som trycktes

`if (event.getButton() == MouseButton.SECONDARY)` – det var höger musknapp som trycktes

⁴ Att koordinater visas tillsammans med namnet beror på att man behöver känna till koordinater för platser för att kunna testa viss annan funktionalitet i inlämningsuppgiften.

Search-operationen förutsätter att man snabbt kan få fram alla platser som har angivet namn. Det behövs en lämplig datastruktur som gör det möjligt. Obs att vi låtsas som om antalet platser kan vara mycket stort, sekvensiell genomgång av olämpliga datastrukturer kan därför inte accepteras.

Hide

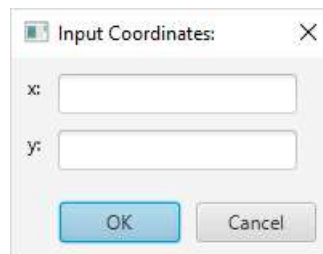
Operationen Hide gömmer alla markerade platser och gör dem avmarkerade. Även denna operation borde stödjas av någon lämplig datastruktur så att man inte behöver gå igenom alla platser utan bara de som är markerade.

Remove

Operationen Remove tar bort alla markerade platser – inte bara så att de inte syns på kartan utan objekten ska tas bort från alla datastrukturer där de kan finnas.

Coordinates

Användaren ska kunna fråga om vad som finns på en viss position på kartan genom att klicka på knappen Coordinates. Detta öppnar en lite dialogruta:



där användaren kan mata in koordinater. Om det finns en plats på dessa koordinater så ska platsen göras synlig (om den var osynlig) och markerad. Eventuella platser som var markerade innan ska avmarkeras. Om det inte finns någon plats på dessa koordinater ska en dialogruta med meddelande om detta visas. I den här dialogen bör det kontrolleras att de inmatade värdena är numeriska.

Obs igen att vi tänker oss att antalet platser är stort, så denna operation behöver också en lämplig datastruktur där man snabbt kan få fram en plats med hjälp av dess position (eller få veta att det inte finns någon plats där). Denna datastruktur behövs även vid skapande av platser, för att kontrollera att det inte redan finns en plats på den klickade positionen.

Kategorier

Platserna kan som sagt tillhöra en av de tre kategorierna (eller vara kategorilösa). Kategorierna visas i listan i högra panelen. Platsen tillhör den kategori som är vald när platsen skapas, om ingen kategori är vald då blir platsen kategorilös. Till varje kategori hör en färg, platsens färg (som triangeln ritas ut i) är kategorins färg. Färgerna är: Bus = `Color.RED`, Underground = `Color.BLUE`, Train = `Color.GREEN`. Kategorilösa platser är svarta.

Meningen med kategorierna är att man snabbt ska kunna gömma eller visa alla platser av en viss kategori, alltså alla T-baneingångar, busshållplatser eller tågstationer.

Om användaren vill gömma alla platser som hör till en viss kategori så väljer hon kategorin i kategorilistan och klickar på knappen Hide category – platser som hör till denna kategori ska göras osynliga. Om användaren vill göra alla platser som hör till en viss kategori synliga så räcker det att markera kategorin i listan.

File-menyn

Programmets data ska kunna sparas på fil⁵ och laddas in vid nästa körning. Operationer för detta finns i menyn File i menybalken. Operationerna i menyn är New Map, Load Places, Save och Exit.

Själva kartbilden ligger på sin fil och ändras inte av programmet, så den behöver inte sparas. Platserna däremot ska kunna sparas (Save) och laddas in (Load Places).

Platserna ska sparas på en textfil, med en plats per rad. På varje rad ska platsens värden skrivas ut i en kommaseparatorerad lista, med platsens typ (Named eller Described), platsens kategori (Bus, Underground, Train eller None om platsen saknar kategori), x-koordinaten, y-koordinaten, platsens namn och (om platsen är av typen Described) dess beskrivning.

Exempel på utseende:

```
Named,Underground,666.0,487.0,Kista
Named,Bus,365.0,235.0,514
Named,Bus,634.0,354.0,514
Named,Underground,450.0,350.0,Husby
Named,Bus,94.0,786.0,179
Named,Bus,762.0,624.0,514
Named,Underground,272.0,197.0,Akalla
Named,Underground,691.0,528.0,Kista
Named,Train,929.0,317.0,Hellenelund
Named,Bus,719.0,507.0,514
Named,Bus,818.0,382.0,514
Named,Bus,527.0,259.0,514
Named,Bus,137.0,717.0,514
Named,Bus,485.0,335.0,514
Named,Bus,883.0,578.0,514
Named,Bus,728.0,511.0,179
Named,Bus,311.0,147.0,514
Named,Bus,760.0,391.0,514
Named,Underground,416.0,313.0,Husby
Named,None,736.0,397.0,NOD
Named,Underground,224.0,169.0,Akalla
Described,None,711.0,453.0,Forum,Där satt DSV till år 2014
Named,Bus,915.0,475.0,514
```

(514 och 179 som namn på platser i ovanstående exempel är bussnummer).

Obs! att det inte får finnas mellanslag kring kommatecken.

Obs! Vid rättning kommer ditt program att testas med våra testfiler, så formatet måste stämma och programmet måste kunna läsa in en sådan fil.

Information om vilka platser som är markerade resp. synliga sparas inte. Vid inladdning av platser är de alla synliga och omarkerade från början.

En sådan testfil gällande kartan jarvafaltet.png finns tillsammans med denna text och heter jarvafaltet.places.

Både Load Places och Save ska visa en filöppningsdialog och fråga användaren om filnamnet där platserna ska sparas/därifrån platserna ska läsas in.

⁵ Filhantering går igenom på föreläsning 15

Exit ska avsluta programexekveringen. Programmet ska även kunna avslutas via stängningsrutan. Om det finns osparade förändringar ska det visas en dialogruta som varnar om att det finns osparade ändringar och frågar om man ändå vill avsluta – användaren har då möjligheten att avbryta operationen.

Obs att både New Map och Load Places kan väljas även när man har en karta inladdad och platser skapade. I så fall måste det aktuella ”dokumentet” avslutas innan det nya kan användas.

Användaren bör ges samma varning som vid Exit i fall det finns osparade ändringar. Dessutom måste alla datastrukturer som innehåller platser tömmas, så att de nya kan skapas/laddas in.

Inlämning

Inlämning ska göras senast 2019-05-29 för rättning i samband med kursen (se nedan för datum för andra inlämningstillfället). Inlämning sker på kurssidan i ilearn2.dsv.su.se.

Det är två filer som ska lämnas in:

- en exekverbar JAR-fil⁶ innehållande körbar version av lösningen
- en fil med namnet `src.zip` innehållande sammanpackade källkodsfiler i ZIP-format

För att ladda upp filerna klicka på ”Inlupp 2 – inlämning” under rubriken ”Inlämningsuppgift 2”.

Klicka på kommentarsfältet för ”Submission comments” och mata in följande information:

- namn och personnummer för samtliga gruppmedlemmar

- vilket operativsystem (Windows, Linux, MacOS) programmet har utvecklats på

Klicka sedan på knappen ”Add submission”. Dra in JAR-filen och ZIP-filen. Klicka på

knappen ”Save changes”. Om uppgiften löstes i grupp är det bara en i gruppen som ska lämna in.

Om du inte hinner bli klar till deadline ovan så kommer ett andra tillfälle den 2019-08-18, med handledning under vecka 33.

Om du inte hinner blir klar med uppgiften till deadline i augusti så upphör uppgiften att gälla och du måste lösa inlämningsuppgifterna på en kommande omgång av kursen. Obs att det gäller samtliga delar av inlämningsuppgiften, alltså även inlupp 1⁷. Nästa omgång av PROG2 ges vårterminen 2020.

⁶ JAR-filer går igenom på föreläsning 18

⁷ Inlupp 1 utgör inte ett eget examinationsmoment utan är en del av examinationsmomentet Inlämningsuppgift.