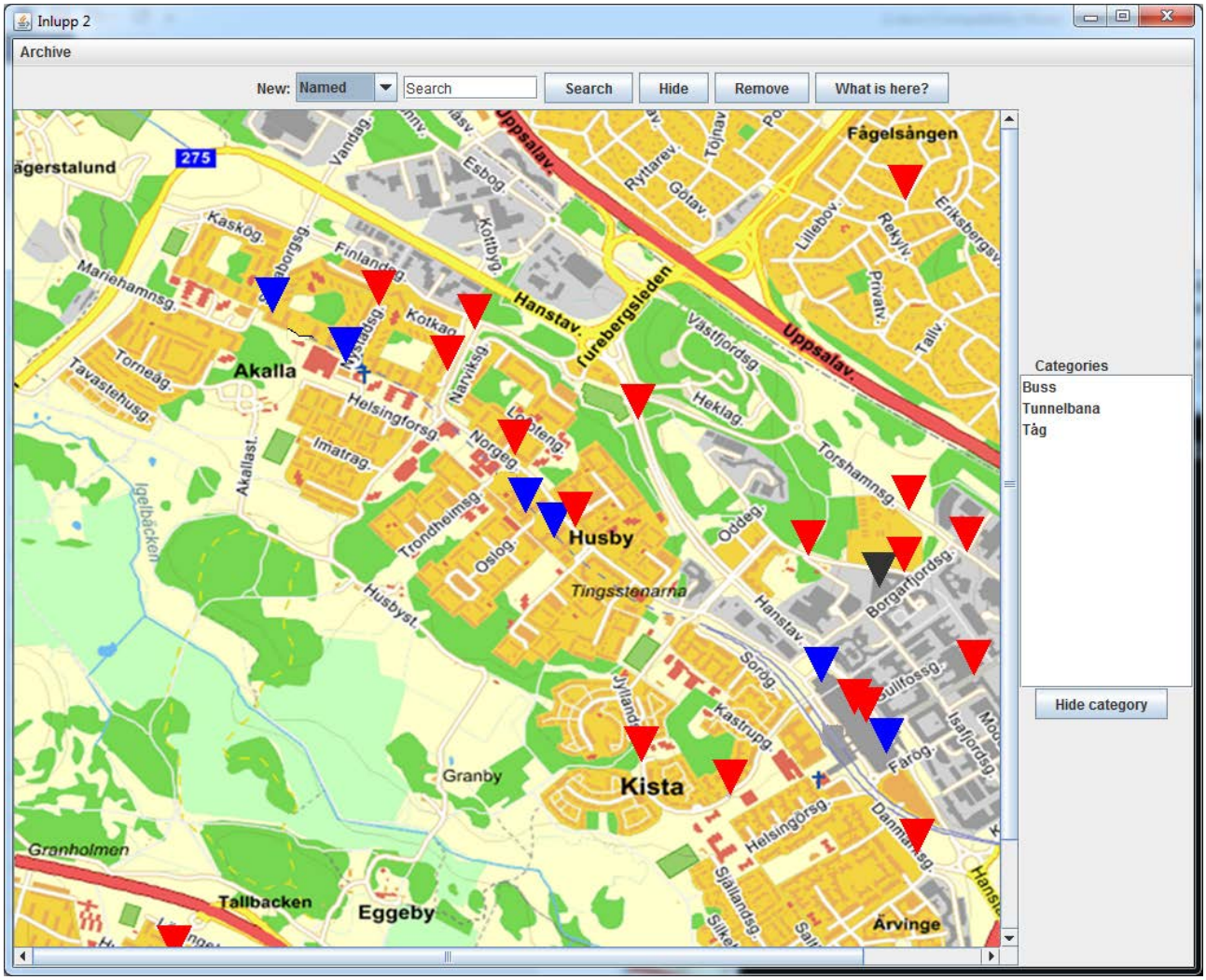
Inlämningsuppgift 2

Denna uppgift är den andra (av två) obligatoriska inlämningsuppgiften på delkursen PROG2 Programmering 2 VT2016. Uppgiften ska lösas enskilt eller i grupper om högst tre personer. En lösningsanvisning och en inlämningsinstruktion finns sist i det här dokumentet.

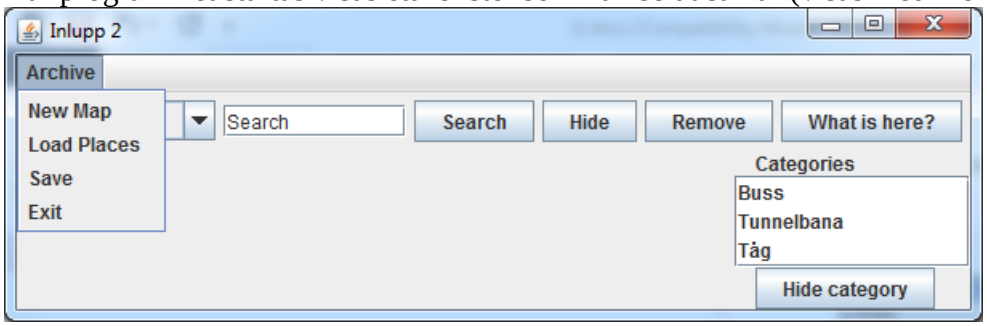
Uppgiften går ut på att skriva ett program som låter användaren öppna en karta (en bakgrundsbild) och med musklickningar definiera platser på kartan. Platserna har namn, kan ha en beskrivande text och kan tillhöra någon av de tre kategorierna Buss, Tunnelbana eller Tåg. Uppgiften är inspirerad av användargränssnittet 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 kartan osv.



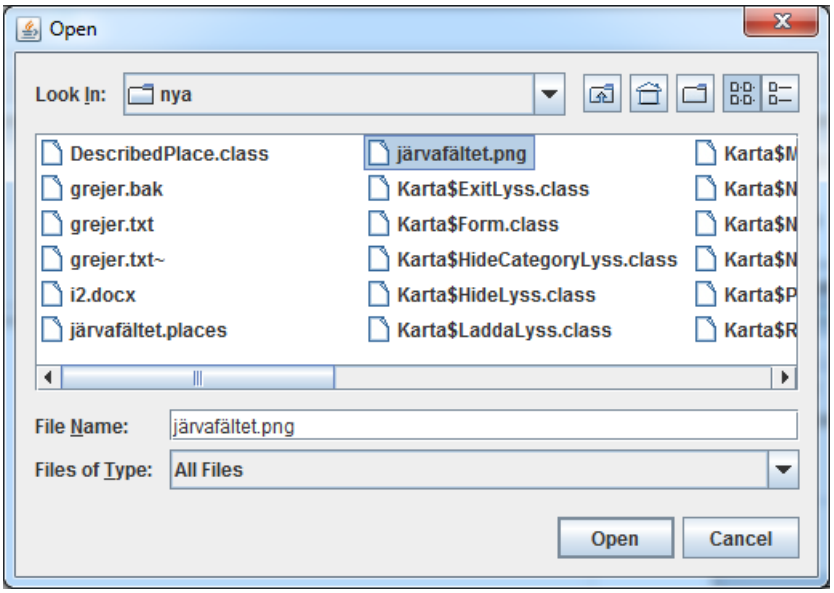
På ovanstående bild är det de färgade och svarta trianglarna som visar platser som användaren har definierat, tanken är att triangelns spets pekar på punkten (pixeln) där platsen är. Färgen på triangeln bestäms av vilken kategori som platsen tillhör, som i sin tur bestäms av vilken kategori som var vald när platsen skapades.

Kartan

Kartan är en vanlig bildfil.
När programmet startas visas ett fönster som kan se ut så här (visas med menyn Archive nerfällt):



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



Bilden ska laddas in och visas i fönstret. Om bilden är större än fönstret ska scrollbalkar visas så att man ska kunna scrolla kartan. Man ska även kunna förstora fönstret genom att dra ut det.
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 Arkiv-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`. 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.[1]

Platser

Platser är tänkta att kunna vara av olika typer, innehållande olika typer av information. I uppgiften ingår att det ska finnas två sorters platser: NamedPlace som bara har ett namn och DescribedPlace som förutom namnet har en beskrivande text (en rad). Namn och beskrivning matas in av användaren när hen skapar platsen och kan inte ändras sedan. Programmet ska vara förberett för att i en framtida utbyggnad ha fler platstyper, t.ex. platser som kan innehålla en bild, en tidtabell,

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

öppettider osv. Det ska alltså finnas en liten klasshierarki för platser, där man lätt kan stoppa in nya platstyper.

En plats skapas genom att användaren väljer kategori i listan till höger och sedan platstypen i comboboxen i övre vänstra hörnet (comboboxen fungerar ju även som en knapp, så det behövs ingen mer knapp för att skapa en plats). Då ska markören över kartan ändras till ett kors (för att markera att nästa klick skapar en plats) och en klick på kartan skapar en plats på den klickade positionen. Obs att det är tänkt att den nedre triangelspetsen ska visa var platsen finns, så det behövs viss justering av koordinater för platsen. Efter att platsen är skapad ska inte kartan vara mottaglig för klickning förrän en platstyp väljs i comboboxen på nytt. Om ingen kategori är markerad när en plats skapas får platsen ingen kategori och dess färg är svart.

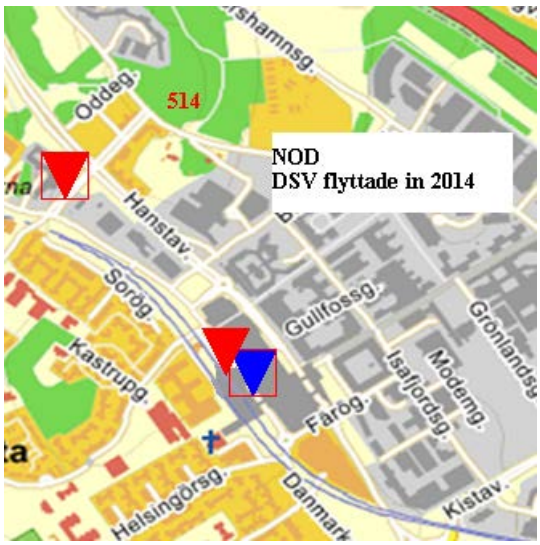
När en NamedPlace skapas ska användaren ange platsens namn, vilket kan ordnas med en JoptionPane.showInputDialog. När en DescribedPlace skapas ska användaren dels ange namnet, dels en beskrivning som (av tekniska skäl) består av bara en rad.

En Plats har tre olika tvåvärdiga tillstånd att hålla reda på: den kan vara hopfälld (visas som en triangel) eller utvikt (visas med sitt namn eller namn och beskrivning), den kan vara markerad eller ej och den kan vara synlig eller gömd (man kan gömma platser man inte är intresserad av för att kartan inte ska bli oöverskådlig).

Hantering av om platsen är synlig eller ej kan implementeras med Swings grafiska komponenters `setVisible`-metoder (Plats-klasser bör givetvis ärva från `JComponent`). Om platsen visas hopfälld eller utvikt resp. markerad eller ej måste implementeras i Plats-klasserna.

Användaren ska kunna markera/avmarkera en plats genom att klicka på den med vänster musknapp[2].
Flera platser ska kunna vara markerade samtidigt. Hur en markerad plats visas kan du bestämma själv, i exemplet nedan till höger är markerade platser omgivna med en röd rektangel (den vänstra röda triangeln och den nedre blå triangeln). En plats ska kunna vara både markerad och utvikt samtidigt.

Man ska kunna vika ut eller fälla ihop platser genom att klicka på dem med höger musknapp. Nedan till vänster är alla platser hopfällda, till höger har den översta röda platsen (tydligen en NamedPlace) vikts ut (texten 514 är tänkt som bussnr på en busshållplats), och den svarta triangeln som markerar NOD-huset (tydligen en DescribedPlace) har vikts ut med namn och beskrivning. En högerklick på de utvikta platserna fäller ihop dem igen.



Obs! att för att kunna rita den utvikta informationen måste man utvidga Plats-komponentens Bounds (med `setBounds`-metoden). När Platsen fälls ihop igen borde Bounds återställas så att

[2] I en MouseListener-metod kan man kolla vilken knapp som trycktes på så här:
if (mev.getButton() == MouseEvent.BUTTON1)… där mev är MouseEvent-argumentet. Det finns fördefinierade namn för BUTTON1, BUTTON2 och BUTTON3.

klickningar med musen utanför triangeln inte markerar platsen.

Obs vidare! att det inte ställs några högre krav på presentation av informationen om platser i utvikt läge. Lägg inte ner tid på det (anpassningar till olika typsnitt och storlekar), bara informationen presenteras så kan detaljer fixas till om tiden tillåter. T.ex. om raden med beskrivande text hos en DescribedPlace är fö lång är det ok att kapa den.

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

Search

Operationen Search hämtar strängen som matats in i sökfältet, letar upp och markerar alla platser som har denna sträng som namn och gör dem synliga och markerade. Obs att flera platser kan ha samma namn. Sökresultatet visas alltså genom att Platserna markeras. Om andra Platser var markerade innan sökningen så avmarkeras de.
Search-operationen förutsätter att man snabbt kan söka upp alla platser som har ett visst namn. Sekvensiell genomsökning av olämpliga datastruktur accepteras inte. Se även lösningsanvisningar på sista sidan.

Hide

Operationen Hide gömmer alla markerade platser och gör dem avmarkerade.

Remove

Oprationen 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).

What is here

Användaren ska kunna klicka på kartan för att få veta om det finns någon osynlig Plats på denna position. Om det finns en plats där så ska den visas, om det inte finns så ska ingenting hända.

Obs att komponenter som inte syns inte reagerar på musklickningar. Det behövs, liksom i samband med Search, någon datastruktur där man snabbt ska kunna få fram om det finns en plats på den klickade positionen eller inte.

Obs vidare att det kan vara väldigt svårt att träffa just den pixeln där den osynliga platsen är. Man ska därför undersöka inte bara den position där användaren har klickat, utan alla positioner i en omgivning till klickpunkten, i en ruta på t.ex. 21x21 positioner där klickpunkten är den mittersta positionen.

Kategorier

Platserna kan som sagt tillhöra en av de tre kategorierna (även om det kan finnas kategorilösa platser). Kategorierna visas i listan i högra panelen. Platsen tillhör den kategori som är vald när platsen skapas, om ingen kategori är vald då så blir platsen kategorilös. Till varje kategori hör en färg, platsens färg (som triangeln ritas ut i i hopfällt läge) är kategorins färg. Färgerna är
Buss = Color.RED, Tunnelbana = Color.BLUE och Tåg = Color.GREEN
Kategorilösa platser är svarta. Meningen med kategorierna är att man ska kunna visa eller gömma alla platser av en viss kategori, alltså alla T-baneingångar, busshållplatser eller tågstationer.

Hantering av platser i kategorier

Platser som hör till en kategori ska visas när användaren markerar den kategorin i listan. Däremot ska platser hörande till en viss kategori inte gömmas när kategorin blir avmarkerad. Vill man gömma alla platser som hör till en kategori så klickar man på knappen Hide Category – platser som hör till den valda kategorin görs osynliga.

Arkiv-menyn

Programmets data ska kunna sparas på en fil[3] och laddas in vid nästa körning. Operationer för detta finns i en menyn Archive i menybalken. Operationerna i menyn är New Map, Load Places, Save och Exit.

Själva kartbilden ligger på sin fil och har inte förändrats, 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 i en kommaseparerad lista, med platsens typ (Named eller Described), platsens kategori (Buss, Tunnelbana, Tåg eller None om platsen saknar kategori), x-koordinaten, y-koordinaten, platsens namn och (om platsen är av typen Described) dess beskrivning. Exempel på utseendet:

```
Named,None,752,390,Macken
Named,Buss,877,580,514
Described,None,731,403,NOD,DSV flyttade in 2014
Named,Buss,763,628,514
Described,None,717,452,Forum,Där bodde vi förr
Named,Buss,423,291,514
Named,Tunnelbana,737,543,Kista
Named,Tåg,931,273,Helenelund
Named,Buss,308,165,514
```

(514 som namn på platser i ovanstående exempel är bussnummer).
Obs att det inte får finnas mellanslag kring kommatecknen.

Informationen 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 `järvafältet.png` finns tillsammans med denna text och heter `järvafältet.places`.
Obs! Vid rättning kommer era program att testas med våra testfiler, så formatet måste stämma och programmet måste kunna läsa in en sådan fil.

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

Exit ska avsluta programmet. Programmet ska även kunna stängas via programmets stängningsruta i övre högra hörnet. Om det finns osparade förändringar ska det visas en dialogruta som varnar om att det finns osparade förändringar 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 förändringar. Dessutom måste alla datastrukturer som innehåller platser tömmas, så att de nya kan skapas/laddas in.

[3] Filhantering gås igenom på föreläsning 15

Lösningsanvisningar

Meningen med uppgiften är att öva dels på GUI med inslag av egenritad grafik, dels på datastrukturer. Platser ska givetvis samlas i någon/några datastruktur/-er.
I en primitiv lösning skulle denna datastruktur 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 mycket stort.

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

För varje sökning eller annan operation bör 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 plaster för varje kategori, motsvarande med platser som har samma namn osv.

Inlämning

Inlämningen ska göras senast 2016-04-29 för rättning i samband med kursen. Inlämning av uppgiften sker på kurssidan i ilearn2.dsv.su.se.
Det som ska lämnas in är en exekverbar JAR-fil[4] innehållande lösningen.
Obs! JAR-filen ska även innehålla en mapp med namnet `sources` innehållande källkoden till lösningen.

För att ladda upp JAR-filen 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. 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 den angivna deadline så kommer ett andra tillfälle den 19 augusti 2016, med handledning under vecka 33.

Om du inte hinner bli klar med uppgiften till deadlinen i augusti så upphör uppgiften att gälla och man ska lösa inlämningsuppgifterna på en kommande omgång av kursen. Obs att detta gäller samtliga delar av inlämningsuppgiften, alltså även inlupp 1[5]. Nästa omgång av PROG2 ges på vårterminen 2017.

[4] JAR-filer gås igenom på föreläsning 17.
[5] inlupp 1 utgör inte ett eget examinationsmoment utan är en del av examinationsmomentet Inlämningsuppgift 4,5hp..