

## Assignments 2

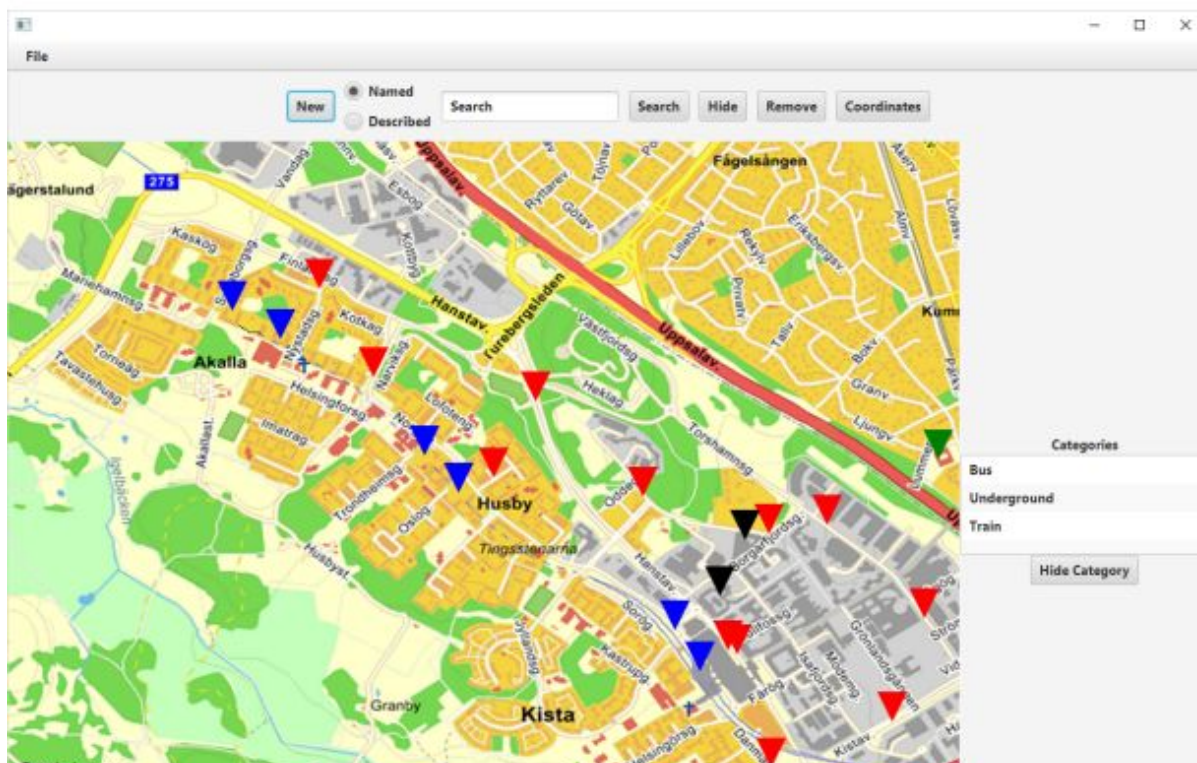
This assignment is the second (of two) compulsory assignments in the *Programming 2* VT2020 module. **The task must be solved individually.**

**Submission instructions can be found in a separate document in iLearn. NOTE: Only Java thus not FXML!**

The task is to write a program that allows the user to open a map (a background image) and to define locations on the map with mouse clicks. These sites can then be manipulated in different ways. **Questions on the assignment are handled on supervision occasions as well as via the course web page**

### Description

places have **names**, *can have a descriptive text* and *can belong to any of the three categories* Bus, Underground or Train<sup>1</sup>, but they can *also be categorical*. The task is inspired by the user interface on [kartor.eniro.se](http://kartor.eniro.se), but is of course very simplified and implemented completely differently: the map is just a picture, there is no zoom, you can not move on the map (it is not larger than shown), etc.



<sup>1</sup> On occasion, you must not rename the categories, because then our tests will not work  
PROG2 VT2020 Assignment 2

In the picture above, the different colored triangles show places that the user has defined. The idea is that the tip of the triangle points to the point (one pixel) where the location is. The color of the triangle is determined by which category (Bus, Underground, Train or *none*) the place belongs to. The category to which a site belongs is determined by which category was selected in the list on the right when the site is created (via the button **New**).

The locations can be of two types: NamedPlace (has only one name) or DescribedPlace (has a name and description). In an intended future expansion one should be able to have more types of places, eg. places with a timetable, places with a picture, etc. *However, these should not be implemented as part of the task.*

Users should be able to perform some operations on the sites: (1) select locations, (2) hide selected locations, (3) delete selected locations, (4) hide or show all locations of a particular category, (5) search for locations with a certain name and (6) check what is on specified coordinates.

The purpose of the task is to practice on the GUI with elements of self-drawn graphics and on the use of the *correct* data structure for a certain problem formulation. We can note that places should be gathered in some data structures. In a primitive solution (ie not the one you are going to use), the data structure could be an ArrayList that you search sequentially (from first to last elements) every time you need to do something with the sites. **However, this is an ineffective solution if the number of seats is large.** See lecture 10 for tips on how to solve this!

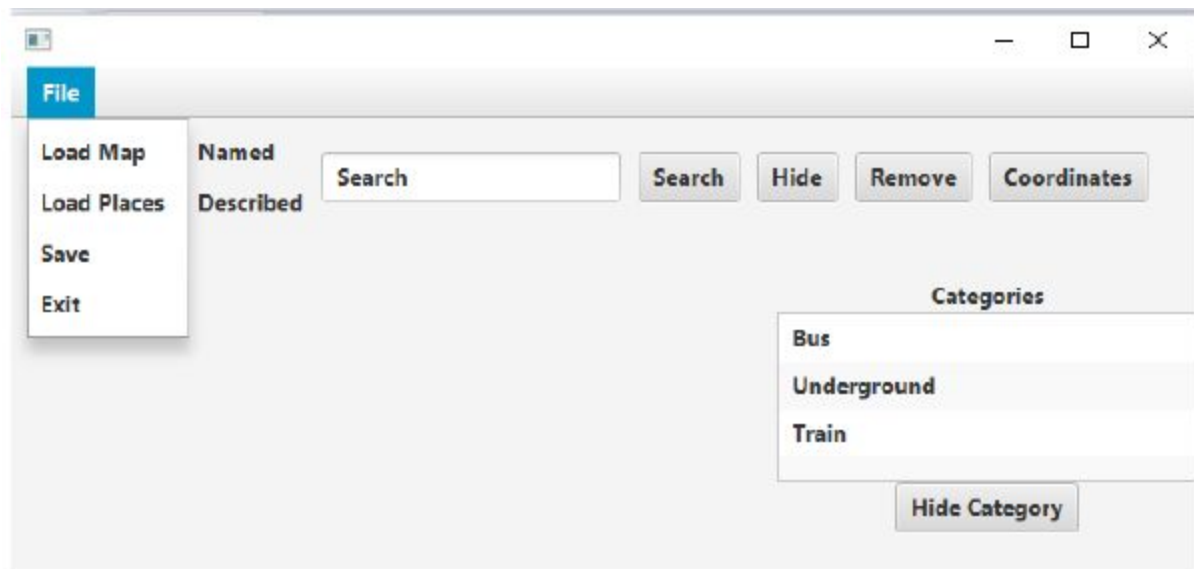
Note that is **not allowed** an over-efficient solution to this task.

*For each search or other operation, think about whether the operation would be facilitated with any particular data structure and if so use this data structure.* For example. When managing categories, there should be a **data structure that collects all locations for each category**, when searching by name, there should be a **data structure that collects all locations with the same name**, and there should also be a **data structure that collects locations by position**. You should design the solution as if the number of seats could be very large.

**Map** The map is a common image file (for example .png, .jpg or .bmp).

PROG2 VT2020 Tutorial 2

When the program starts, a window will look like this (shown here with the menu **File** down):



If you select **Load Map** , a file dialog will be displayed where the user can select an image (hopefully displaying a map): The

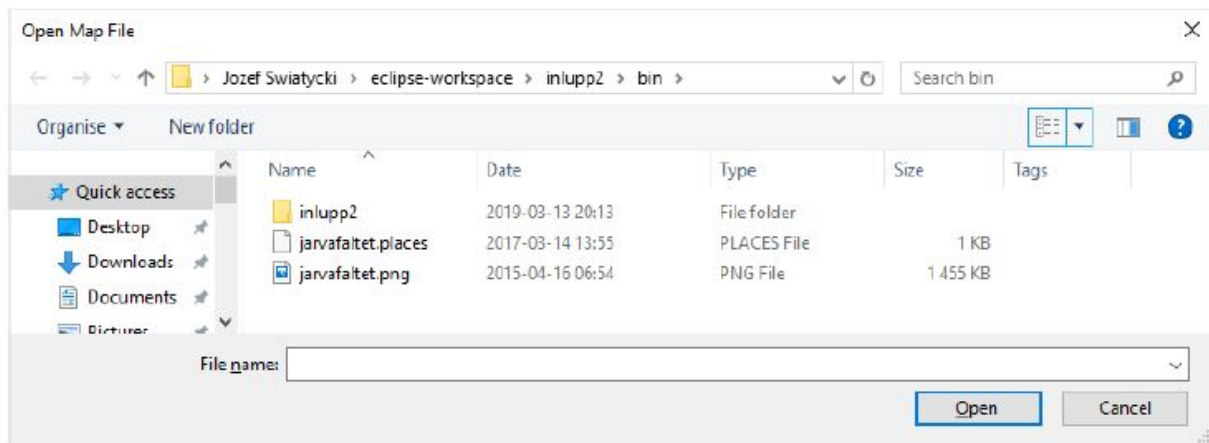


image should be loaded entered and displayed in the window. You should be able to enlarge the window by dragging it out so you can see a larger part of the map - the **map should not be scaled back then**.

**Note** Along with this task text is the file jarvafaltet.png which contains an image with a map of Norra Järva. There is also a file jarvafaltet.places with saved locations (see **File** menu below) that applies to this particular map.

**Position** Each place has a name that is specified when the place is created, a position and a color (as determined by the category). The position simply specifies the pixel coordinates of the location and should be represented by a property class named Position. Objects of this class thus enclose only two coordinates, x and y, represented as two **int**.

A Position should be used to identify locations when you want to know what lies in a particular position. The position class should therefore be prepared so that its objects will function well as

PROG2 VT2020 Hand-in assignment 2

keys in hash tables.<sup>2</sup>

Note the small complication that the site's graphic area is counted from its upper left corner, while the location's position according to this application is the triangle's lower point. *This requires some recalculation of coordinates.*

**Places** Places are thus meant to be of different types, containing different types of information. An object-oriented solution to this is to create a class hierarchy, where the different place types are subclasses to a superclass Place. The assignment includes two types of locations: NamedPlace which (in addition to coordinates, category etc.) has only one name and a DescribedPlace which (in addition to coordinates, category etc.) has both a name and a descriptive text.

A location is created by the user selecting the category in the list on the right, selecting the location type using the radio buttons at the **New** button and pressing the button **New**. Then the cursor over the map should be changed to a cross (to mark that the next click on the map creates a location) and a click on the map creates a location on the clicked position. *Note that it is intended that the lower triangle tip shows where the location is, so some adjustment of the coordinates of the location is needed.* If no category is selected when a site is created, the site becomes categorical and its color becomes black.

If there is already a location at the clicked point, an error message should be given - only one location per position is allowed. See the description of Operation Coordinates below for some more information on this.

When creating a NamedPlace, the user must enter the place name. When creating a DescribedPlace, the user must specify the name and a description that (for technical reasons) consists of only one line.

A place should be **visible or hidden** (you can hide places you are uninterested in to make the map more clear) and it can be **marked or not**. *In addition, the user should be able to right-click on a site and then see its information in a dialog.*

Handling of whether the site is visible or cannot be implemented with calls to the setVisible method of graphical components (The Place class must of course inherit from any subclass

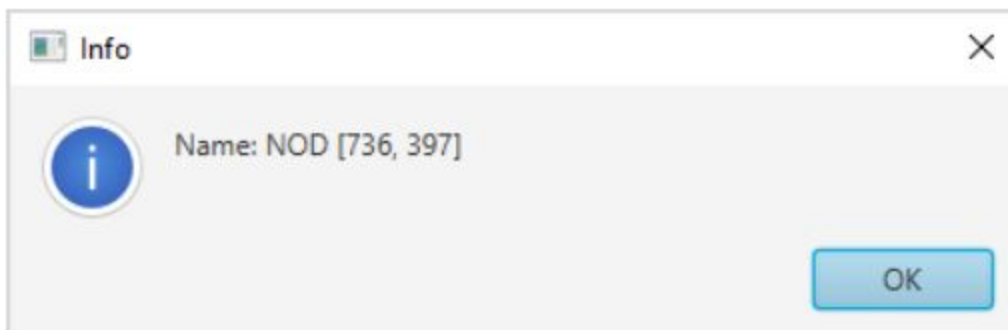
to Node). Whether the place is selected or not should be implemented in the Place class.

The user should be able to select or deselect a location by clicking on it with the left mouse button<sup>3</sup>. Several places should be able to be marked at the same time. You can decide how a selected place is displayed, for example marked places can be shown in the color yellow and with black edges.

The user should be able to click on a site with the right mouse button and then be able to see information about the site. The information is displayed in a dialog box. The information displayed depends on the location type: for a NamedPlace, the name and coordinates are displayed<sup>4</sup>:

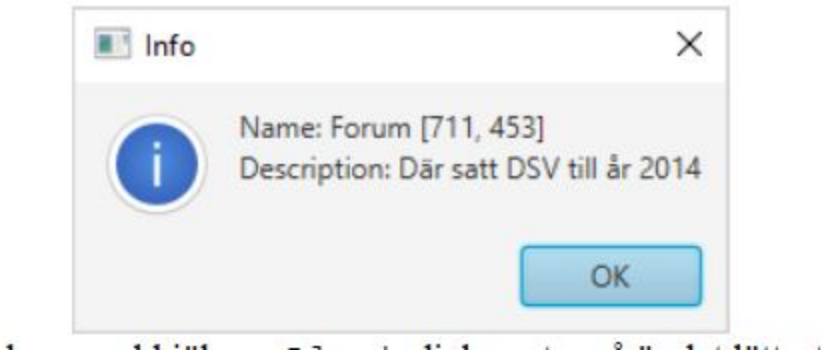
<sup>2</sup> There is a similar class Point in Javas library, but for practice it is compulsory to write your own such class instead. <sup>3</sup> In a mouseEvent handle method, you can see which button was pressed like this: if (event.getButton () == MouseButton.PRIMARY) - it was the left mouse button that was pressed if (event.getButton () == MouseButton. SECONDARY) - it was the right mouse button that was pressed

<sup>4</sup> That coordinates are displayed together with the name is because you need to know the coordinates of places in order to test some other functionality in the assignment task.



For a DescribedPlace, the name is displayed with coordinates, and the description:





If this display is arranged using Alert dialogs, it is easiest for the information to be displayed only for one place at a time - this is a permitted restriction in the task. An error message can thus be given if several places are selected and the user wants to view information about a place.

The program must be able to keep track of which locations are selected - the **Hide** and **Remove operations** work in the **selected** locations.

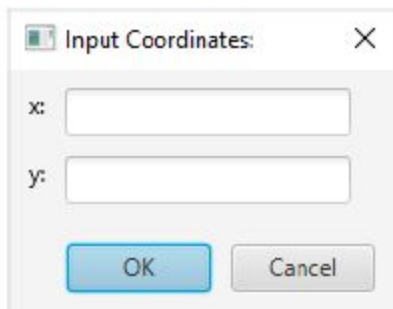
**Search** operation **Search** locates places using the name entered into the search field. It starts by unchecking places that may have been marked before the search. It then retrieves the search string from the search field, making all sites that have that name **visible and highlighted**. *The result of the search is thus presented by places with this name being highlighted and visible.*

**The Search** operation assumes that you can quickly find all the places that have the given name. A suitable data structure is needed to make this possible. *Note that we pretend that the number of sites can be very large, sequential review of inappropriate data structures can therefore not be accepted.*

**Hide** surgery **Hide** hides all selected sites and makes them unchecked. Also, this operation should be supported by some suitable data structure so that you do not have to go through all the locations but only those that are marked.

**Remove** operation **Remove** removes all selected sites. *The location should be removed from all data structures where they can be found, so they should not only be hidden so that they are not visible on the map.*

**Coordinates** **The user** should be able to ask about what is in a certain position on the map by clicking on the button **Coordinates**. This opens a little dialog:



The image shows a standard Windows-style dialog box. The title bar reads 'Input Coordinates:' followed by a close button (X). The main area contains two text input fields, the first labeled 'x:' and the second labeled 'y:'. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

where the user can enter coordinates. If there is a place on these coordinates then the place should be made **visible and highlighted**. *Any places that were marked before should be cleared.* If there is no space on these coordinates, a message dialog box should be displayed. In this dialog it should be checked that the values entered are numeric.

Note again that we think the number of places is large, so this operation also needs a suitable data structure where you can quickly get a place using its location (or find out that there is no place there). *This data structure is also needed when creating locations, to check that there is not already a location at the clicked position.*

**Categories** As mentioned, the places can belong to one of the three categories (or be

category-free). The categories appear in the list in the right panel. The site belongs to the category that is selected when the site is created, if no category is selected then the site becomes categorical. Each category has a color, the color of the place (in which the triangle is drawn) is the color of the category. The colors are: **Bus** = Color.RED, **Underground** = Color.BLUE, **Train** = Color.GREEN. *Categoryless places are black*. The purpose of the categories is to be able to quickly hide or show all places of a certain category, ie all subway entrances, bus stops or train stations. **Implement category as a class** with name, color and possibly other methods and instance variables.

If the user wants to hide all places that belong to a particular category, select that category in the category list and click on the button **Hide category**. Places that belong to the selected category should be rendered invisible. *If the user wants to make all the places that belong to a certain category visible, it is sufficient to select the category in the list.*

**File menu** The program data should be able to be saved on file<sup>5</sup> and loaded at the next run. Operations for this can be found in the menu **File** in the menu bar. The menu operations are **New Map**, **Load Places**, **Save** and **Exit**.

The map image itself is in its file and is not changed by the program, so it does not need to be saved. The locations however, will be saved(**Save**)and loaded(**LoadPlaces**).

The locations should be saved to a text file, with one space per line. On each row, the values of the place should be printed in a comma-separated list, with the type of the place (Named or Described), the category of the place (Bus, Underground, Train or None if the place has no category), the x-coordinate, the y-coordinate, the place name and ( if the site is of the type Described) its description.

Examples of appearance:

```
Named, Bus, 485,335,514 Named,  
Underground, 666,487, Kista Named,  
Bus, 311,147,514  
Named,Bus,634,354,514  
Named,Underground,450,350,Husby  
Named,Bus,762,624,514  
Named,Bus,365,235,514  
Named,Underground,691,528,Kista  
Named,Train,929,317,Hellenelund  
Named,Bus,719,507,514
```

Named,Bus,818,382,514  
Named,Bus,94,786,179  
Named,Underground,224,169,Akalla  
Named,Bus,527,259,514  
Named,Bus,883,578,514  
Named,Bus,728,511,179  
Named,Underground,416,313,Husby  
Named,Bus,760,391,514  
Named,None,736,397,NOD  
Named,Bus,137,717,514  
Named,Underground,272,197,Akalla  
Described,None,711,453,Forum,Där satt DSV till år 2014  
Named,Bus,915,475,514

<sup>5</sup> File management is reviewed during

(514 and 179 as names of places in the above example are bus numbers). **Note that there must be no space around commas. Note If corrected, your program will be tested with our test files, so the format must match and the program must be able to load such a file.**

Information on which places are marked or resp. visible is not saved. When loading locations, they are all visible and unchecked from the start.

Such a test file concerning the map jarvafaltet.png is found with this text and is called jarvafaltet.places.

Both **Load Places** and **Save** should display a file opening dialog and ask the user if the file name where the locations should be saved / from which the locations should be loaded.

**Exit** should end the program execution. The program should also be able to be terminated via the closing box. If there are unsaved changes, a dialog box should be displayed that warns that there are unsaved changes and asks if you still want to quit - the user then has the option to cancel the operation.

Note that both **New Map** and **Load Places** can be selected even when you have a map loaded and places created. In this case, the current "document" must be completed before

the new one can be used. The user should be given the same warning as at Exit in case of unsaved changes. In addition, all data structures that contain locations must be emptied so that the new ones can be created / loaded.