Assignment 3: A New World

Overview

This assignment will introduce you into the world of linked data structures, using an example of alternative city planning. Your task will be to build houses in an area, but to create a different infrastructure between the houses than you know from your usual city area. Let's first think about how the development of undeveloped landscapes takes it form in usual cases: some landscape architect defines roads (let's for simplicity assume a single road), allocates lots along the road, assigns a sequence of numbers to these lots. What the buyer can choose is the location, yet restricted to a lot.

In a computer, such an approach could be modeled using an array, the street-name is the array-name, the lot number is the index. Such a system has many advantages: you know exactly where the data is stored (or where the house is built), and if you want to access the data (e.g. pick someone up from the house), it's easy to find. However, the access and freedom of choosing the location is quite restricted: you have to stick with your lots (and if there are no more lots available, you need to plan a new road!).

Here is an alternative access to this problem: no landscape architect. No roads. You have your area, and you can build your house wherever you want to. The only condition you need to fulfill, in order to stay connected to the world, is to go to the most recently built house, and tell the owner where you proudly place your property. The very first settler in this area needs to put his/her location (=address) onto a sign-post at the (defined) entrance of that area. Btw., there are no roads. You need a horse or 4WD truck in that area.

Let's look how you would visit every single house in the area: you would start at the sign-post. Read the address. Remember it. Let's call this the "current" address. From there on, you repeat the following task (do you sense a while loop here?):

- ride to the current address.
- Ask the owner about the next address (he/she knows, because his successor-settler had to tell him!)
- Make this your current address (do you sense a JAVA code line "current = current.next" here?)

you do this, until the owner says "there is no other house left".

Let's see what you would have to do to link your own house into this line:

- select a space, build your house.
- go to the sign-post, find the first house. Visit all houses, until you found the last one. Leave your new address at that house.

Now, there's some advantage to this kind of settlement: you can build your house wherever you want to. You only have to stop when the area is full, not when there's no road to it! However, it is a bit harder to find your house, e.g. for mail-delivery: the poor mail-man/woman has to successively visit each house, starting from the first settlement (which is remarked on the sign-post), until he/she finds you.

You see, there are ups and downs to structures. This is what we will talk about in class. However, your task for now is to simulate the alternative landscape situation.

Task Details

You will visualize the landscape using SimpleGUI. The area to develop is you SimpleGUI screen.

Task #1: place 10 houses at random places. On building each house, you have to link it to the previously built one. A house can be as simple as a box, or you can design it more in detail. The link to the previous house must be drawn, too, this means, you must draw a line between a house and its successor.

Task #2: on keypress, visualize the path of visiting every house, i.e. visit each house (one house per keypress), such that the currently visited house changes its color.

Hints: From Task to Implementation

This linked structure has a lot to do with objects and their references. Imagine a house is represented by an instance of a class "House" (this is a requirement. You MUST design a class house). This class will contain a method that visualizes a house (e.g., "drawHouse"), a method that visualizes the connection to the following house, and it will have to reserve some place to store the address of the following house. Think for a second what that is: the address to the following house is a reference to, yes, a House-object. Hence, your class must contain a field of type House, which, when newly instantiated, will be null, but then can carry the address to the next house when required (i.e. when the next house is built). Remember, this field of type House is in the class definition of class House --- this can be confusing, but it's totally fine: of course a class can contain a reference of an object of its own type.

You then want to build a class that actually represnts the area, and which contains the methods that are related to the area, e.g. visit all houses (which could also be called, traverse the area's houses in the order they were added), add a house etc. In this class, you also want to have a field variable that represents the sign-post. Of course, the sign post is of type.... yes, House. It refers to a house. It's a reference to a House-object.

Requirements

Visualization must be done using SimpleGUI

- At least 3 classes have to be created: Main, Area, House. Main contains the main logic. Area contains methods to deal with the houses, "add", "visitNext". Do NOT put these methods into the class "House"! House contains a reference to the next house ("House next;"), and a method to draw the house, incl. the connection to a following house, if there is one.
- you get 5 points if you are able to build and connect 10 houses at random places on the screen. The houses must be internally connected, i.e. when a new house is built, there MUST be a text output (console or SimpleGUI): "new house built at <X,Y>, connected to house at <X,Y>". The first house needs to write "First house built. connected to sign post".
- You get 10 points, if, after building all houses, you can visually traverse them. This means, you show all 10 houses, e.g. in red. Each time the key "t" is pressed, the next house (starting with the first one built), is re-colored in green.

Bonus points: bonus points for animation: have a postman going from house to house! Make beautiful houses! Add trees! And a complicated one: make sure, two houses don't overlap! Make the area clickable, i.e. the houoses are not built at random locations, but where you place them! Give the house a steaming chimney! Add squirrels!

Good luck.