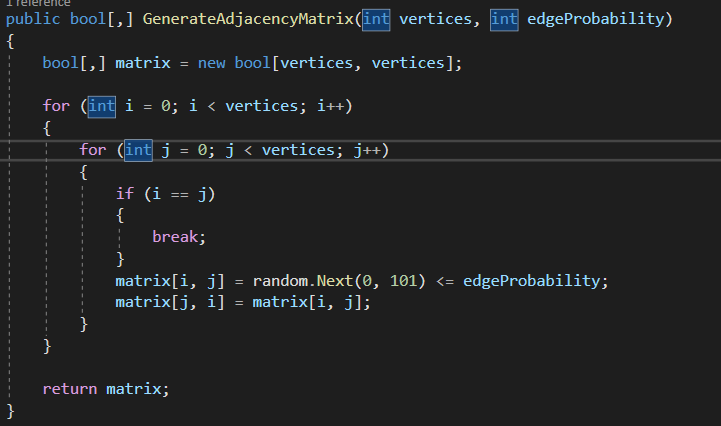
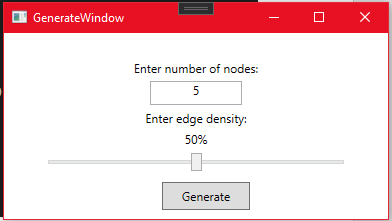
Week 1

# Generating the adjacency matrix

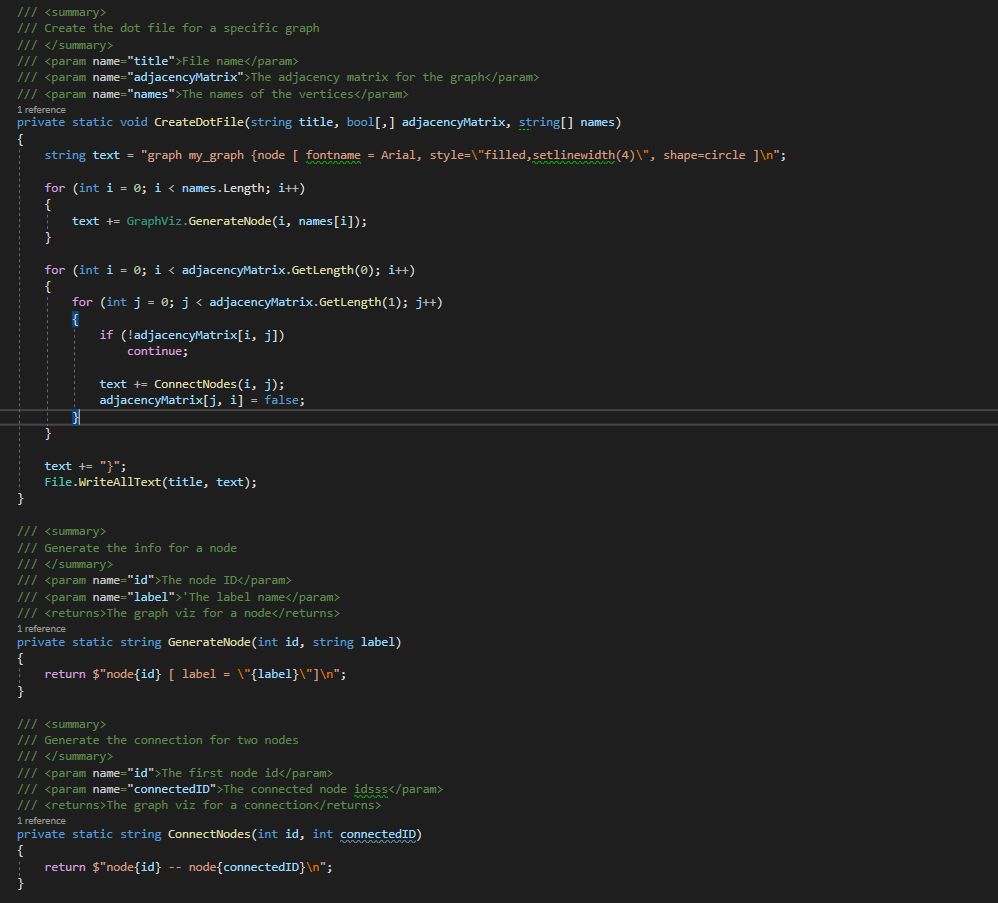
To generate the adjacency matrix we first loop trough half of the adjacency matrix. If the random value we generate is above the specified possibility of creating an edge we add this to the adjacency matrix. We then also add this value the oppositive value. This is done with this code:



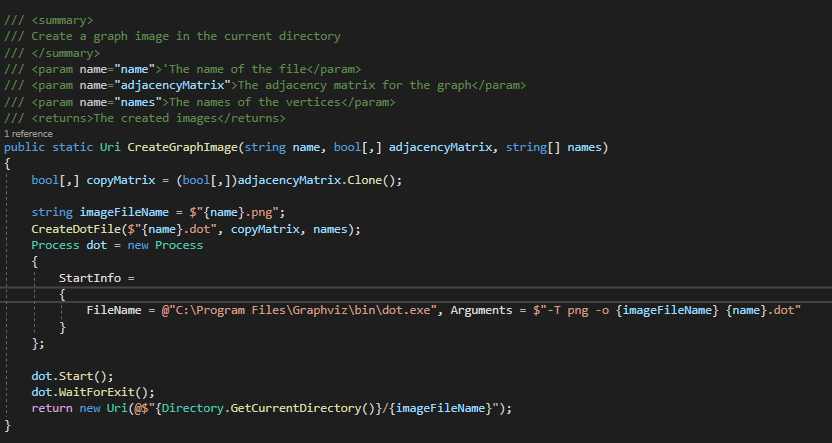
To get the size and probability of an edge we created this small pop up window:  


# Graph viz

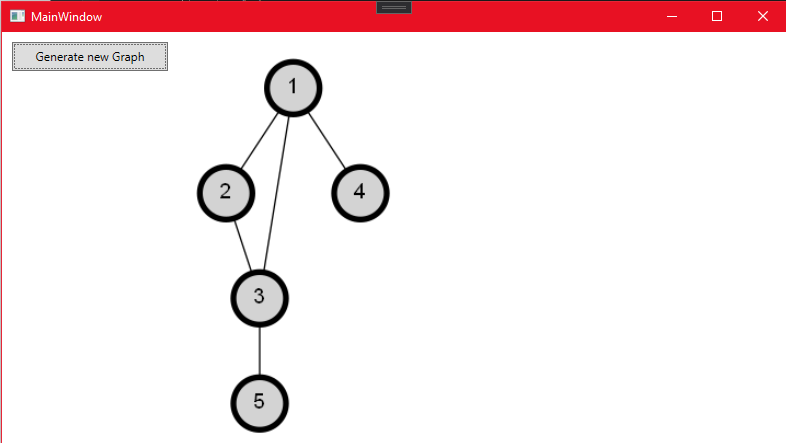
After generating the adjacency matrix we had to create an image of its corresponding graph. We did this by creating a small script that generates a simple dot file. The dot file is used by Graph viz. This is the code that we used to generate the dot file:



Then we pass this specific dot file to Graph Viz. We do this by using the C# processing system:



Then we can load in the image into our basic win form. This looks like this:



# Connecting the graphs

For connecting the graphs, we wrote a small algorithm that finds the set of all sub-graphs within the parent-graph.

We do this by putting all the nodes of the graph in a sequential list. Then, for each node we create a sub-graph structure to which we add the current node. We then check in the adjacency matrix to which other nodes the current node is connected and add those to the sub-graph as well (provided we have not traversed them already). We also remove said nodes from the sequential list.

We then repeat this process of adding connected nodes to the sub-graph and removing them from the sequential list, until no other nodes can be added to the sub-graph. At this point we add the current sub-graph to the list of sub-graphs and move on to the next node, for which we run through the above process again. We repeat this until there are no other nodes left in the sequential list.



When we have a set of sub-graphs, we can use these to connect them together in the adjacency matrix. For this we wrote another simple method that simply takes the first nodes of to sub-graphs and connects them together and repeating this for every subgraph (except the last one). We then finally end up with a graph that is fully connected.

