DESIGN DOCUMENT

GROUP MEMBERS:

NGOC KIEU THANH HUYNH – 2688093

BILAL BUTT – 2688700

RAIMA KHAN –2692686

ARMIN ROSHAN- 2487128

CONTENTS

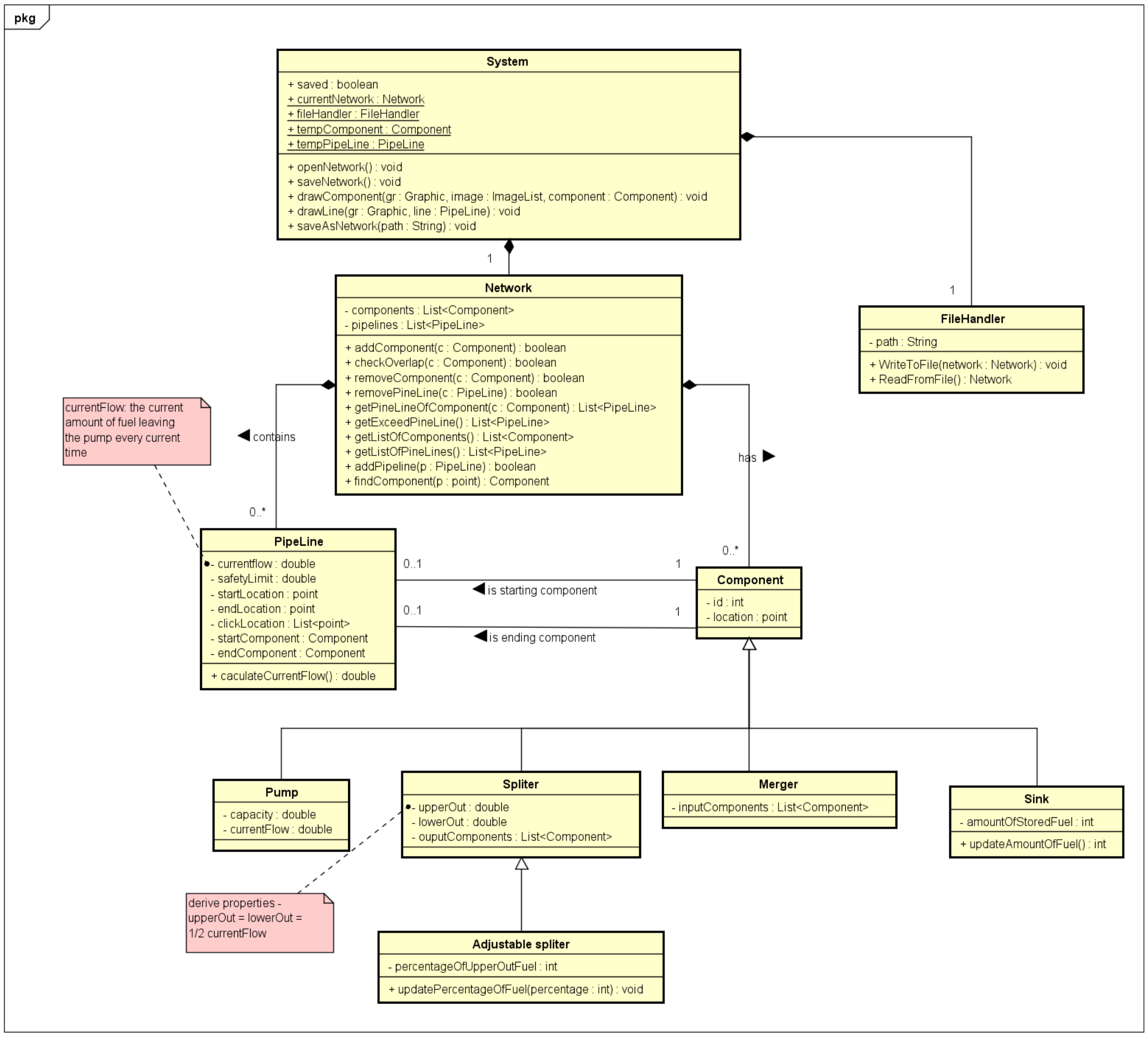
INTRODUCTION

This document is the design document for building an application that simulates an oil pipe-line network.

In the first section of this document, the class diagram of the application are presented in the form of diagrams along with their brief descriptions. The second section presents selected sequence diagrams for some of the more complex and important methods related to various classes.

CLASS DIAGRAM

CLASS DIAGRAM



pkg
System
+ saved : boolean
+ currentNetwork : Network
+ fileHandler : FileHandler
+ tempComponent : Component
+ tempPipeLine : PipeLine
+ openNetwork() : void
+ saveNetwork() : void
+ drawComponent(gr : Graphic, image : ImageList, component : Component) : void
+ drawLine(gr : Graphic, line : PipeLine) : void
+ saveAsNetwork(path : String) : void
1
Network
- components : List<Component>
- pipelines : List<PipeLine>
+ addComponent(c : Component) : boolean
+ checkOverlap(c : Component) : boolean
+ removeComponent(c : Component) : boolean
+ removePineLine(c : PipeLine) : boolean
+ getPineLineOfComponent(c : Component) : List<PipeLine>
+ getExceedPineLine() : List<PipeLine>
+ getListOfComponents() : List<Component>
+ getListOfPineLines() : List<PipeLine>
+ addPipeline(p : PipeLine) : boolean
+ findComponent(p : point) : Component
1
FileHandler
- path : String
+ WriteToFile(network : Network) : void
+ ReadFromFile() : Network
currentFlow: the current amount of fuel leaving the pump every current time
contains
0..*
has
0..*
PipeLine
- currentflow : double
- safetyLimit : double
- startLocation : point
- endLocation : point
- clickLocation : List<point>
- startComponent : Component
- endComponent : Component
+ caculateCurrentFlow() : double
0..1
1
is starting component
0..1
1
is ending component
Component
- id : int
- location : point
Pump
- capacity : double
- currentFlow : double
Spliter
- upperOut : double
- lowerOut : double
- ouputComponents : List<Component>
Merger
- inputComponents : List<Component>
Sink
- amountOfStoredFuel : int
+ updateAmountOfFuel() : int
derive properties - upperOut = lowerOut = 1/2 currentFlow
Adjustable spliter
- percentageOfUpperOutFuel : int
+ updatePercentageOfFuel(percentage : int) : void

DESCRIPTION OF THE CLASSES AND THEIR MEMBERS

COMPONENT CLASS

Component class is the parent class, represented the generic properties for all type of components.

Properties		
ID	Integer	This is unique number to indicate the components of each network drawing. It is increased automatically when the component is added
Location	Point	Representing an ordered pair of integer x- and y-coordinates that defines a location of components on the screen
Operations		
Component(id: int, location: Point): *constructor of the class*		

PUMP CLASS

Pump class is the child class of Component class. It has the following characteristics:

Properties		
Capacity	double	A value of 'Capacity' indicates the capacity of Pump's flow.
CurrentFlow	double	A value of 'CurrentFlow' indicates the current flow of the Pump.
Operations		
Component(id: interger, location: point, capacity: double, currentflow: double*): constructor of the class*		

SINK CLASS

Sink class is the child class of Component class. It has the following characteristics:

Properties		
AmountOfStoredFuel	integer	A value of this field indicates the current flow of the Sink.
Operations		
UpdateAmountOfFuel()	integer	This method updates the flow of the Sink and returns the value of new flow.
Component(StFuel: int): constructor of the class		

MERGER CLASS

Splitter class is the child class of Component class. It has the following characteristics:

Properties		
InputComponents	List<Component>	A list of input components used by Splitter class.
Operations		
Component(): constructor of the class		

SPLITTER CLASS

Splitter class is the child class of Component class. And parent class of Adjustable Splitter class. It has the following characteristics:

Properties		
upperOut	double	A value of this field indicates the flow of splitter's upper terminal.
lowerOut	double	A value of this field indicates the flow of splitter's lower terminal.
outputComponents	List<Component>	A list of output components used by Splitter class.
Operations		
Component(upOut: double, loOut: double): constructor of the class		

ADJUSTABLE SPLITTER CLASS

Adjustable Splitter class is the child class of Splitter class. It has the following characteristics:

Properties		
PercentageOfUpperOutFuel	integer	A value of this field indicates the percentage of splitter's upper flow.
Operations		
UpdatePercentageOfFuel(int percentage)	void	This method updates the percentage of splitter's flow on lower terminal.
Component(perUpper: int): constructor of the class		

PIPELINE CLASS

Properties		
currentFlow	Double	Representing the current flow of the pipeline
saftyLimit	Double	Representing the maximum safety limit of the pipeline
startLocation	Point	Indicating the starting location of the pipeline
endLocation	Point	Indicating the ending location of the pipeline
clickLocation	List<Point>	Indicating the points in the middle of the pipeline
startComponent	Component	The component containing the starting location
endComponent	Component	The component containing the ending location
Pipeline(currentflow: double, limit: double, startLocation: Point, endLocation: Point, clickLocation :List<Point>) : *constructor of the class*		
caculateCurrentFlow(): double *This method is used to calculated and update the current flow of the pipeline whenever the other pipelines are added. Base on the startCompnent and endComponent, it's possible to calculate the values of currentflow and update it.*		

NETWORK CLASS

The Network class in the class hold main objects of the application.

Properties		
components	List<Component>	A list of Component objects, private and are accessible through getListOfComponent() methods
pipeLines	List<Pipeline>	A list of Pipeline objects, private and are accessible through getListOfPipeLine() methods
Network(components : List<Component>, pipeLines: List<PineLine>) : *constructor*		
+ addComponent(c : Component) : boolean *This method is used to calculated and update the current flow of the pipeline whenever the other pipelines are added. Base on the startCompnent and endComponent, it's possible to calculate the values of currentflow and update it.*		
checkOverlap(c : Component) : Boolean		

Check whether the given component c is overlap any other components in the list of components
removeComponent(c : Component) : Boolean *Find the given component c and remove it from the list of componets. When c is removed, the connected pipeline of c also have to be removed.* *Return true if c is found and removed*
removePineLine(c : PipeLine) : Boolean *Find the given pipeline from parameter list and remove it.* *Return true if c is found and removed*
getPineLineOfComponent(c : Component) : List<PipeLine> *Get the list of pipeline connected to the given component c.* *Return the list of pipeline*
getExceedPineLine() : List<PipeLine> *Check for the exceed pipelines (if the currentflow > safetylimit) and return into a list of pipelines*
getListOfComponents() : List<Component> *Return the private object **components***
getListOfPineLines() : List<PipeLine> *Return the private object **pinelines***
addPipeline(p : PipeLine) : Boolean *Add a given pipeline p into the list of pipelines, which is represented by object **pipelines**.* *P is added if and only if the location of p is not overlapped other components in the list. Method **checkOverlap** is called for this check. If it is not overlap, it is added into the list and this method return true.*
findComponent(p : point) : Component *Find that weather there are a component having the location which contain s the given point or not. If it is true, return the found component. Otherwise, return null.*

SYSTEM CLASS

The System class is the main class in our class heirarchy, which is a static class and has two properties of type Network and FileHandler. This class is the main class in our application and uses the two objects in its property.

Properties		
saved	Boolean	Keep track the state of the current network drawing.

		Saved is true if there are not any new changes. Otherwise, it is false.
currentNework	Network	The object of network class, indicating the current network that the user are working on.
fileHandler	FileHandler	Handling functions relating to save, open network drawing
tempComponent	Component	These two properties help the System verify and know which component the user is trying todraw or whether the user is trying to draw a pipe-line.
tempPipeline	Pipeline	
openNetwork() : void *Open the network drawing from the text file based on fileHandler.path and fileHandler.ReadFromFile()*		
saveNetwork() : void *Save the network drawing into the text file based on fileHandler.path and fileHandler.ReadFromFile()*		
saveAsNetwork(path : String) : void *Save as the network into a text file for the first time working with it. The path is indicated by the users and assigns to fileHandler.path*		
drawComponent(gr : Graphic, image : ImageList, component : Component) : void *Draw the given component on the drawing screen depending on which button from the toolbox the user has selected*		
drawLine(gr : Graphic, line : PipeLine) : void *Draw the given pipeline on the drawing screen.*		

FILEHANDLER CLASS

The FileHandler Class is one of the most important classes in the application. It handles functionalities relating to files.

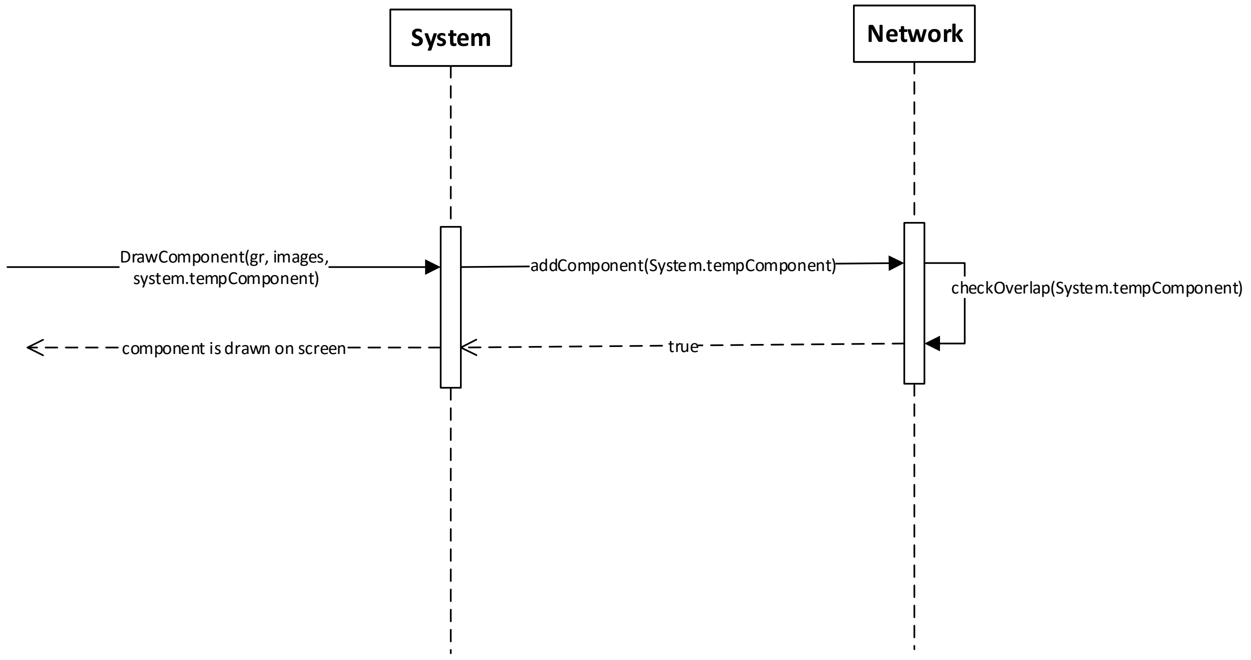
Properties		
Path	String	The path property indicates the file path where the text file storing the Network objects related information is stored.
WriteToFile(network : Network) : void *The WriteToFile takes a Network object as a parameter, writes the list of components and pipe-lines from the lists of the given Network object to a text file in a pre-determined convention for writing such objects to file.*		

ReadFromFile() : Network

The ReadFromFile method reads a text file that has components and pipelines written to it in a pre-determined convention and then takes those components and pipe-lines and assigns them to a Network object which it returns.

SEQUENCE DIAGRAMS

DRAW A COMPONENT


System
Network
DrawComponent(gr, images, system.tempComponent)
addComponent(System.tempComponent)
checkOverlap(System.tempComponent)
component is drawn on screen
true

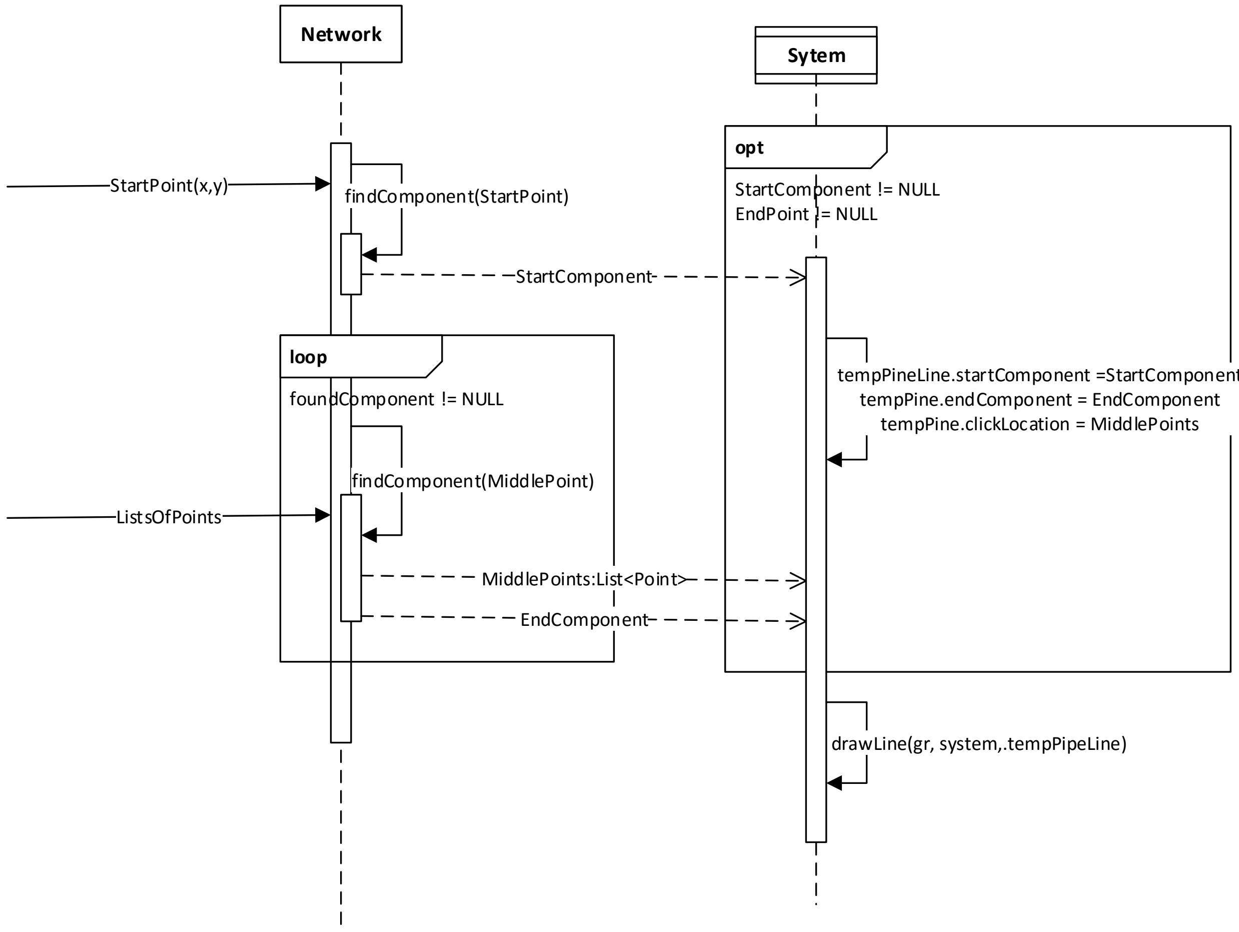
This sequence diagram describe how a component to the drawing screen and add that component to the list of components belonging to the object currentNetwork. The drawComponent method has three argument parameters which include objects of type Graphic, ImageList and Component. The Graphic object can come from the Form's PaintEventArgs or otherwise by creating the Graphic object of the form or a control. The ImageList consists of the images for each of the images for the components that the user would like to place on the drawing screen.

The component object comes from the temComponent property of the System class. This property changes its reference to a new Component object whenever the user clicks on a button for different components on the toolbox. Clicking on a specific component button would create a Component of that type and change the tempComponents reference to that newly created Component object. The component argument in the method is then used to call the method addComponent on the currentNetwork property of the static System class.

The method addComponent takes an argument of type Component which would be the same as tempComponent in this case. The method addComponent then adds the Component to the List of Components property of Network class. However when the addComponent method is called there is another method call within this method to the method for checkOverlap method which would return a boolean. If the return value is true then the component would be successfully added to the list of components.

Once the component is successfully added to the list of components the Graphic object of the method will use the location property of the component to draw the image from the ImageList corresponding to the component that needs to be drawn on the drawing screen.

DRAW A PIPELINE


Network
Sytem
opt
StartComponent != NULL
EndPoint != NULL
StartPoint(x,y)
findComponent(StartPoint)
StartComponent
loop
foundComponent != NULL
tempPineLine.startComponent =StartComponent
tempPine.endComponent = EndComponent
tempPine.clickLocation = MiddlePoints
findComponent(MiddlePoint)
ListsOfPoints
MiddlePoints:List<Point>
EndComponent
drawLine(gr, system,.tempPipeLine)

This sequence diagram for drawing a pipeline. The drawPipeLine method takes two argument parameters, a Graphics object type and a PipeLine object. The PipeLine object is created whenever the click on pipe-line button in form event is raised. In this event the tempPipeLine property of System class is assigned a new PipeLine object without that PipeLine object having any start and end Component properties assigned.

The next time whenever the user clicks on the screen the event for screeen click goes through various checks that first include to check if the tempPipeLine is empty and if not the nested statements check for the following: whether the startComponent of the tempPipeLine is null and assigns a startComponent to it via the method findComponent, if startComponent is not null and endComponent is null and calling the method findComponent returns a null then the point from the EventArgs is added to that tempPipeLines list of clickLocations and finally if startComponent is not null and endComponent is null and calling the method findComponent returns a non-null value then the startComponent is assigned whichever component is returned by findComponent function unless that component is the same as the startComponent.

Once the PipeLine object given in the drawPipeLines argument is complete in that it at least has a start and end Components that are non-null, then the Graphic object is used to draw the PipeLine according to the startComponent and endComponent locations as well as the locations given in the list of clickLocations of the PipeLine object from the parameter.