|  |
| --- |
| Fontys University of applied sciences |
| Design Document |
| Final version |
|  |
| 9/12/2015 |
|  |

*Rosen Danev*

*Preslav Gerchev*

*Dimitar Vikentiev*

*Monica Stoica*

Table of Contents

[Introduction 3](#_Toc437441829)

[Class diagram 4](#_Toc437441830)

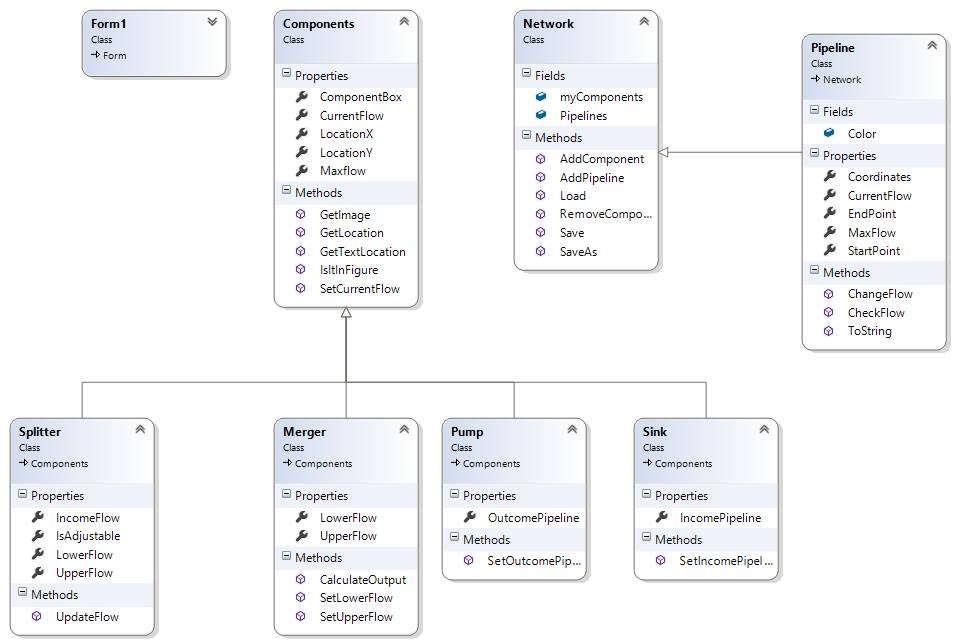
# Introduction

The purpose of this document is to identify the design of our network software system. The system’s structure will be defined using the Unified Modeling Language .

The first chapter will be represented by a class diagram and the description of each class’ members such as fields, properties and methods. The class diagram provides an overview of the software system by describing the classes inside the system and the relationship between them.

Moreover, to have a better understanding of how the objects interract with others in a paricular scenario (use-case), few sequance diagrams will be explained.

# Class diagram



Below you can find a description of the fields, properties and methods used in our class diagram.

***Component –***The base abstract class, parent toall the components except Pipeline. Contains the common properties and methods.

*Private fields:*

* locationX, location –storing the upper left point of the component box

*Properties:*

* ComponentBox – returns the Rectangle that contains the component (and the image). Used to check if it intersects with other components when they are added to our network.

*Methods:*

* *Void GetImage() –* returns the image that will be used for every component. It’s an abstract method and must be overridden in every derived class with the path to the corresponding picture (they are saved in Resources and can be accessed via Properties.Resources.NameOfTheComponent).
* *Point* GetLocation() – returns a Point,created off the coordinates of locationX and locationY. Used to draw the image on this Point.
* *Point* GetTextLocation() – returns a Point where the text that contains the current flow will be drawn.
* Boolean CheckOverlapComponent(Component otherComponent) – returns a Boolean, used to verify if there any other figures that intersect with the componentBox.
* Boolean UpdateFlow() – Used to notify the component that a pipeline has been attached and to update the component’s state.

**Splitter –** A derived class from ***Component***.

*Private Fields:*

- lowerHalf and upperHalf – of type Rectangle – registering the upper and lower half of the component’s box. It is used when splitting the incoming pipeline

*Properties*:

* IncomingPiepline – Type pipeline – The pipeline that is coming into the splitter
* LowerOutcomePipeline- Type pipeline- The lower pipeline that goes out of the splitter
* UpperOutcomePipeline - Type pipeline- The upper pipeline that goes out of the splitter
* IsAdjustable – returns a Boolean. If the splitter is adjustable or not.

*Methods:*

* Rectangle GetHalfOfComponent(Point p) – returns either the lower or the upper half based on point p
* SetLowerOutcomePipeline(Piepline p)- used to set the lower income pipeline
* SetUpperOutcomePiepline(Piepline p) – used to set the upper income pipeline
* SetIncomingPipeline(Piepline p) – used to set the income

**Merger –**A derived class from ***Component.***

*Private Fields:*

- lowerHalf and upperHalf – of type Rectangle – registering the upper and lower half of the component’s box. It is used when mergin the incoming pipelines

*Properties*:

* OutcomePiepline – Type pipeline – The pipeline that is going out of the merger
* LowerIncomePipeline- Type pipeline- The lower pipeline that goes in the merger
* UpperIncomePipeline – Type pipeline- The upper pipeline that goes in the merger

*Methods:*

* Rectangle GetHalfOfComponent(Point p) – returns either the lower or the upper half
* SetLowerIncomePipeline(Piepline p)- used to set the lower income pipeline
* SetUpperIncomePiepline(Piepline p) – used to set the upper income pipeline
* SetOutcomingPipeline(Piepline p) – used to set the outcome

**Pump** – A derived class from ***Component.***

*Properties:*

* OutcomePipeline – a property of type Pipeline that references to the pipeline that comes from that pump.

*Methods:*

* SetOutcomePipeline() – sets the outcoming pipeline.

**Sink** – A derived class from ***Component.***

*Properties:*

* IncomePipeline – a property of type Pipeline that references to the pipeline that comes into that pump.

*Methods:*

* SetIncomePipeline() – sets the incoming pipeline.

**Pipeline**

*Private field*:

- String color – used to mark the color of the pipe. Red if the current flow is equal to the max flow. (is it a string or a Color?)

- inBetweenPoints – a list of Points that contains all the points that are in between the source and the destination of the pipeline. Used when the pipeline is being drawn.

*Properties:*

* CurrentFlow – the current flow of the pipeline.
* MaxFlow- the max flow of the pipeline.
* StartPoint- the point where the pipeline starts.
* EndPoint – the point where the pipeline ends.

*Methods*:

* ChangeCurrentFlow() – used to change the CurrentFlow.
* Boolean CheckFlow() – used to check if the current flow does not exceed the max flow or if it is equal( and change the color of the pipeline accordingly).
* ToString – returns the flow of the pipeline to string

**Network**

*Properties:*

* MyComponents – a list of all components that are currently located on the picturebox.
* Pipelines – a list of all piplines that are currently located on the picturebox.

*Methods:*

* AddComponent() – adds a component to the list.
* AddPipeline() – adds a pipeline to the list.
* RemoveComponent() – removes a component from the list.
* RemovePipeline() – remove a pipeline
* Load() – loads a file for the user.
* SaveAs() – saves the file for the user for future use.
* Save() – automatically saves the file if it has already been saved once before.