|  |
| --- |
| Fontys University of applied sciences |
| Design Document |
| Final version |
|  |
| 10/12/2015 |
|  |

*Rosen Danev*

*Preslav Gerchev*

*Dimitar Vikentiev*

*Monica Stoica*

Table of Contents

[Introduction 3](#_Toc437539865)

[Class diagram 4](#_Toc437539866)

[Sequence Diagrams 8](#_Toc437539867)

[1. Remove component (splitter) 8](#_Toc437539868)

[2. Save file 8](#_Toc437539870)

[3. Change the flow of the pump 9](#_Toc437539871)

[4. Add component (merger) 9](#_Toc437539872)

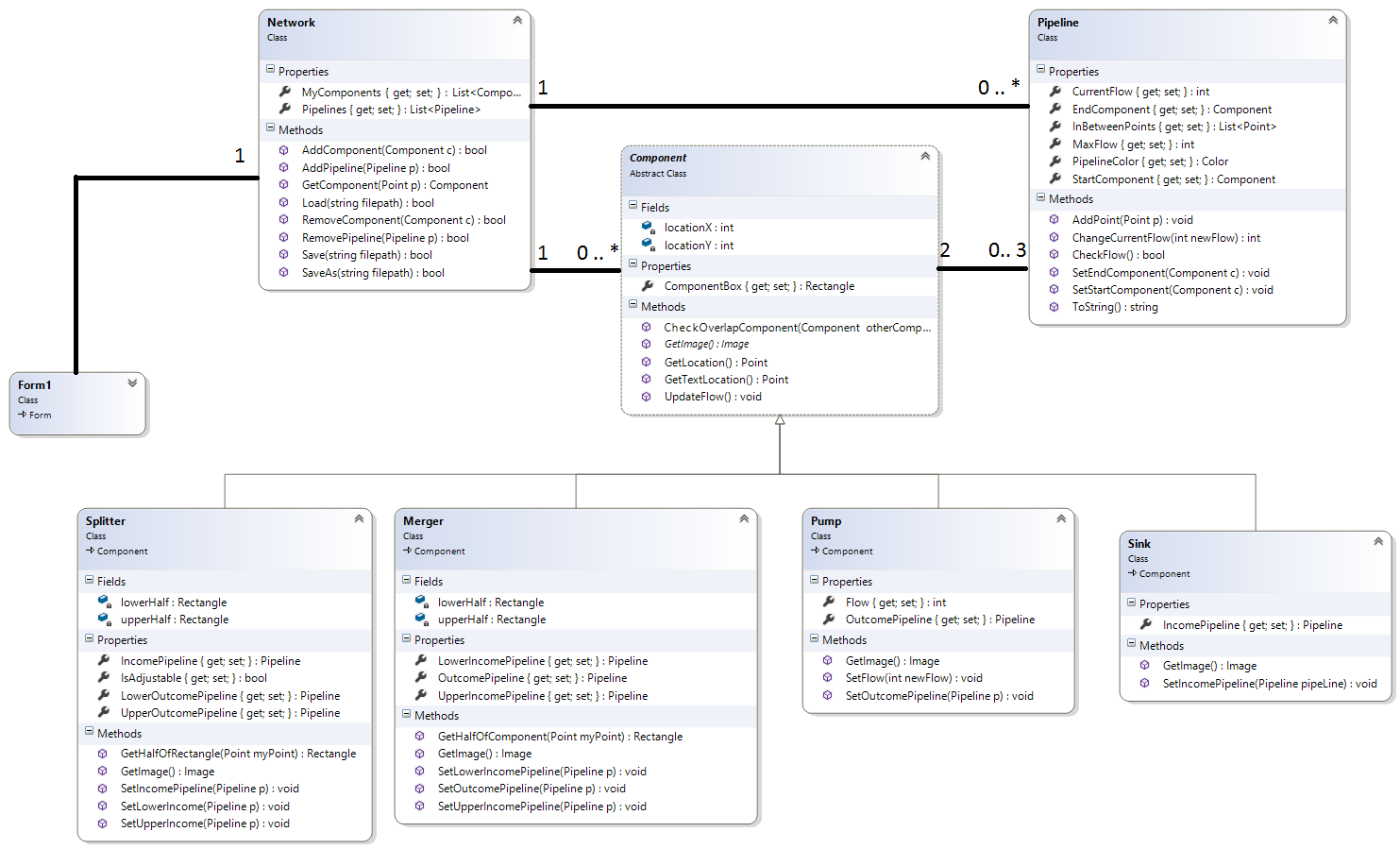
# Introduction

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 Modelling 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 interact with others in a particular scenario (use-case), few sequence diagrams will be explained.

# Class diagram



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

**NOTE: All the properties have a *private set****.* **We were unable to make Visual Studio show that.**

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

*Private fields:*

* locationX : int – stores the X value of the upper left point of the component box.
* locationY: int – stores the Y value of the upper left point of the component box.

*Properties:*

* ComponentBox: Rectangle– 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:*

* *GetImage():Image –* returns the image that will be used for every component. It is 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).
* GetLocation(): Point– returns a Point,created off the coordinates of locationX and locationY. Used to draw the image on this Point.
* GetTextLocation(): Point – returns a Point where the text that contains the current flow will be drawn.
* CheckOverlapComponent(Component otherComponent):bool – returns a Boolean, used to verify if there any other figures that intersect with the componentBox.
* UpdateFlow() : void– 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 : Rectangle – returns the lower half of the Splitter’s ComponentBox.  
Used when drawing the lower outgoing pipeline.  
-upperHalf: Rectangle– returns the upper half of the Splitter’s ComponentBox.  
Used when drawing the upper outgoing pipeline.

*Properties*:

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

*Methods:*

* GetHalfOfComponent(Point p):Rectangle – returns either the lower or the upper half based on the passed parameter.
* SetLowerOutcomePipeline(Pipeline p):void- used to set the lower income pipeline.
* SetUpperOutcomePipeline(Pipeline p):void – used to set the upper income pipeline.
* SetIncomePipeline(Pipeline p):void – used to set the income pipeline.

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

*Private Fields:*

- lowerHalf : Rectangle – returns the lower half of the Merger’s ComponentBox.  
Used when drawing the lower incoming pipeline.  
-upperHalf: Rectangle– returns the upper half of the Merger’s ComponentBox.  
Used when drawing the upper incoming pipeline.

*Properties*:

* OutcomePipeline: Pipeline- The pipeline that is going out of the merger.
* LowerIncomePipeline: Pipeline- The lower pipeline that goes in the merger.
* UpperIncomePipeline: Pipeline- The upper pipeline that goes in the merger

*Methods:*

* GetHalfOfComponent(Point p):Rectangle – returns either the lower or the upper half based on the passed parameter.
* SetLowerIncomePipeline(Pipeline p):void - used to set the lower income pipeline.
* SetUpperIncomePipeline(Pipeline p):void –used to set the upper income pipeline.
* SetOutcomePipeline(Pipeline p):void – used to set the outcome pipeline.

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

*Properties:*

* OutcomePipeline: Pipeline- the pipeline that comes from that pump.
* Flow: int– the Flow that the Pump produces. It is the current and the max flow at the same time.

*Methods:*

* SetOutcomePipeline(Pipeline p):void – sets the outcoming pipeline.
* SetFlow(int newFlow):void – sets the new value for Flow property.

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

*Properties:*

* IncomePipeline: Pipeline- the pipeline that comes into that pump.

*Methods:*

* SetIncomePipeline(Pipeline p):void – sets the incoming pipeline.

**Pipeline**

*Properties:*

* CurrentFlow:int – the current flow of the pipeline.
* MaxFlow:int- the max flow of the pipeline.
* InBetweenPoints:List<Point> – 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.
* StartComponent:Component - the component where the pipeline starts.
* EndComponent:Component – the component where the pipeline ends.
* PipelineColor:Color – returns the color of the pipeline,accordingly to the current flow.

*Methods*:

* ChangeCurrentFlow(int newFlow):void – used to change the CurrentFlow.
* CheckFlow(): bool – 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).
* SetStartComponent(Component c):void – sets the StartComponent property.
* SetEndComponent(Component c):void – sets the EndComponent property.
* AddPoint(Point p): void – Adds a point to the InBetweenPoints list.
* ToString():string – returns the flow of the pipeline to string

**Network**

*Properties:*

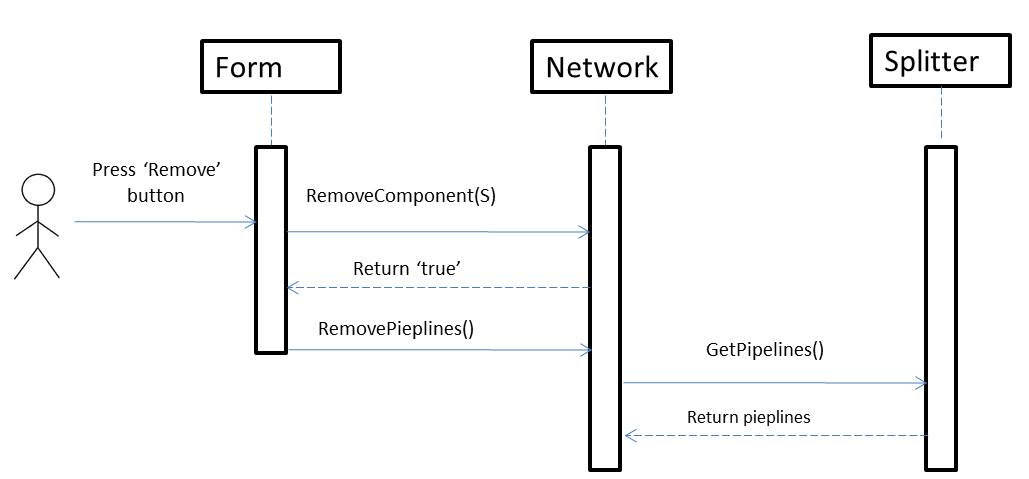
* MyComponents:List<Component> – a list of all components that are currently located on the picturebox.
* Pipelines:List<Pipeline> – a list of all pipelines that are currently located on the picturebox.

*Methods:*

* AddComponent(Component c):bool – adds a component to the list. Returns true if the component has been successfully added, otherwise false.
* AddPipeline(Pipeline p):bool – adds a pipeline to the list.. Returns true if the pipeline has been successfully added, otherwise false.
* RemoveComponent(Component c):bool – removes a component from the list. Returns true if the component has been successfully removed, otherwise returns false.
* GetComponent(Point p):Component – returns the component which is on the location, specified by the parameter
* RemovePipeline(Pipeline p):bool – removes a pipeline from the list of pipelines. Returns true if the pipeline has been successfully removed, otherwise false.
* Load(string filePath):bool – loads a file for the user. Returns true if the file has been loaded successfully, false otherwise.
* SaveAs(string filePath):bool – saves the file for the user for future use. Returns true if the file has been saved successfully, false otherwise.
* Save(string filePath):bool – automatically saves the file if it has already been saved once before. Returns true if the file has been saved successfully, false otherwise.

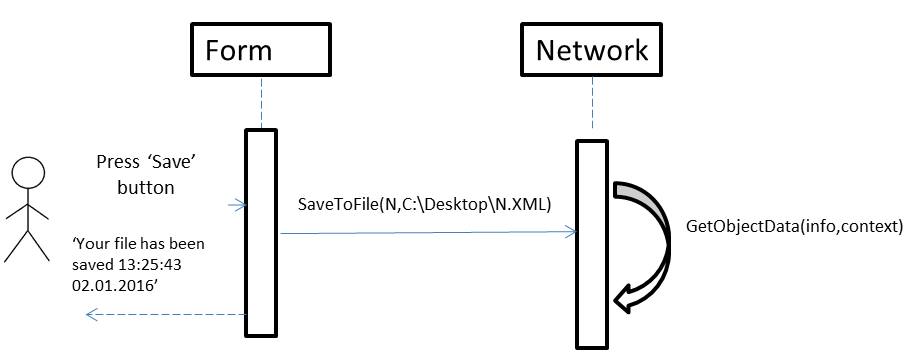
# Sequence Diagrams

### Remove component (splitter)



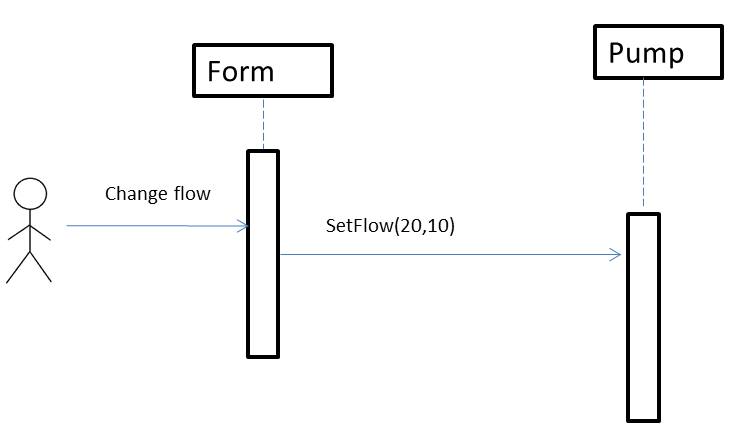
### When the user wants to remove a component, he selects the component and presses the button Remove. Afterwards, the component is deleted from the system as well as the pipelines connected to it, if there are any

### Save file



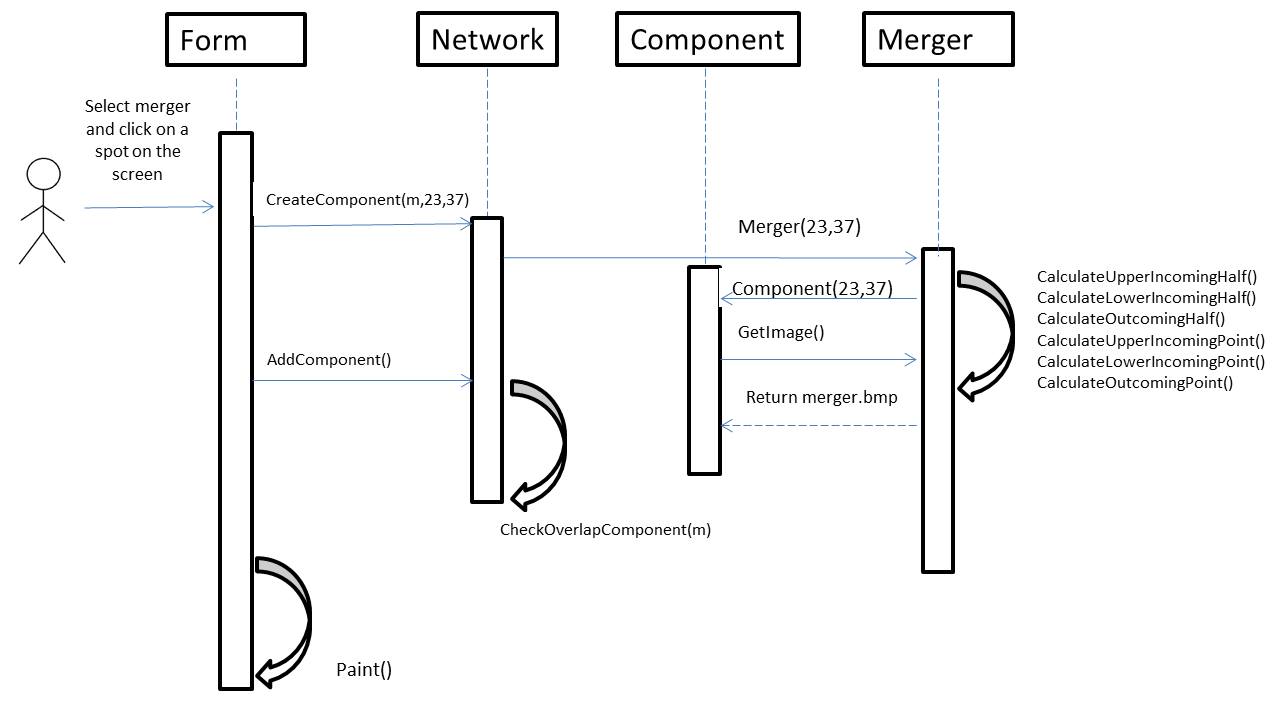
When saving a file, the system will convert the objects to binary. Afterwards, an informative message will be shown on the screen.

### Change the flow of the pump



The user selects the pump and inserts the new flow. The system gets the type of the component based on the location and the method SetMaxFlow() changes the internal flow value (and capacity) of the pump.

### Add component (merger)



To add a component, select the component from the toolbox and clicking on a spot on the screen. The system will create the component with the specified type and on the specified point (x and y coordinates), add the component to the after checking it against other components location to make sure it does not overlap, and getting the proper image. When the component is added to the system then it will be drawn on the screen.