|  |
| --- |
| Fontys University of applied sciences |
| Design Document |
| Final version |
|  |
| 09/01/2016 |
|  |

*Rosen Danev*

*Preslav Gerchev*

*Dimitar Vikentiev*

*Monica Stoica*

Table of Contents

[Introduction 3](#_Toc440135831)

[Class diagram 4](#_Toc440135832)

[Sequence Diagrams 11](#_Toc440135833)

[1. Remove component (splitter) 11](#_Toc440135834)

[2. Save file 11](#_Toc440135835)

[3. Change the flow of the pump 12](#_Toc440135836)

[4. Add component (merger) 12](#_Toc440135837)

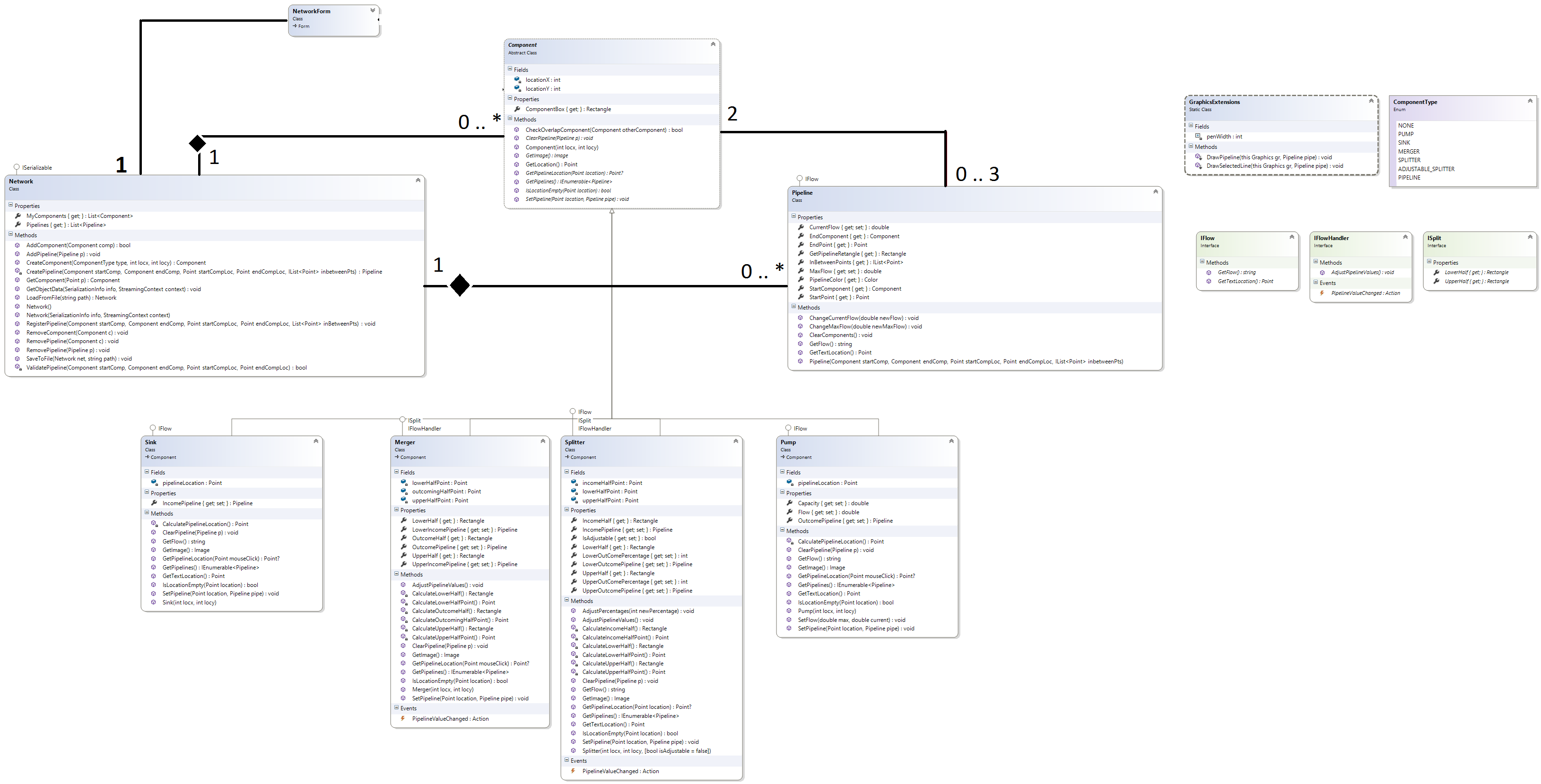
# Introduction

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.

A class diagram and the description of each class’ members such as fields, properties and methods will represent the first chapter. 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.

*Constructor:*

* Component(int x, int y) - creating the component with the specified coordinates.

*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).
* ClearPipeline(Pipeline p): Void – Deletes the pipeline and sets the corresponding outcome/income of the component to null.
* GetPipelines (): IEnumerable<Pipeline> - returns a list with all the connected pipelines.
* SetPipeline(Point location, Pipeline pipe): void – Assigns the pipeline that is being passed based on the location of the mouse click.
* GetPipelineLocation(Point location): Point – Returns the location of the pipeline.
* IsLocationEmpty(Point location) – Checks if the pipeline where the mouse click has been made is null or not.
* GetLocation(): Point– returns a Point, created off the coordinates of locationX and locationY. Used to draw the image on this Point.
* CheckOverlapComponent(Component otherComponent): bool – returns a Boolean, used to verify if there any other figures that intersect with the componentBox.

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

*Private Fields:*

- lowerHalfPoint : Point

-upperHalf Point: Point

-incomeHalfPoint: Point

*Properties*:

* 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.
* IncomeHalf: Rectangle– returns the incoming half of the Splitter’s ComponentBox.Used when drawing the upper outgoing pipeline.
* 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 depending if the splitter is adjustable or not.
* UpperOutcomePercentage:Int – initial value 50
* LowerIncomePercentage:int –initial value 50

*Constructor:*

* Splitter(int locx, int locy, bool isAdjustable=false):base(locx, locy)

*Methods:*

* CalculateUpperHalf():Rectangle – Calculates and returns the upper half of the out coming rectangle
* CalculateLowerHalf():Rectangle – Calculates and returns the upper half of the out coming rectangle
* CalculateIncomingHalf():Rectangle – Calculate sand returns the upper half of the out coming rectangle
* CalculateUpperHalfPoint(): Point – Calculates and returns the point that will be used for the pipeline connection of the upper half.
* CalculateLowerHalfPoint(): Point – Calculates and returns the point that will be used for the pipeline connection of the lower half.
* CalculateIncomingHalPointf(): Point – Calculates and returns the point that will be used for the pipeline connection of the incoming half.
* GetHalfOfComponent(Point p):Rectangle – returns either the lower or the upper half based on the passed parameter.
* GetFlow():String – Returns a string with the percentage of the upper out coming pipeline.
* AdjustPercentage(int newPercentage): void – Sets the percentage of the upper out coming pipeline of an adjustable splitter.
* AdjustPipelineValues(): void – Part of the IFlowHandler interface. Called when the Pipeline value is raise and updates all the values of the connected pipelines.
* GetTextLocation(): Point – Returns the text location for the upper out coming pipeline ‘s percentage.

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

*Private Fields:*

lowerHalfPoint : Point

-upperHalf Point: Point

-outcomingHalfPoint: Point

-PipelineValueChanged:Action event

*Properties*:

* 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.
* OutcomeHalf: Rectangle– returns the outcome half of the Merger’s ComponentBox.Used when drawing the upper incoming pipeline.
* 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

*Constructor*

Merger(int locx, int locy): base(locX, locY)

*Methods:*

* CalculateUpperHalf():Rectangle – Calculates and returns the upper half of the out coming rectangle
* CalculateLowerHalf():Rectangle – Calculates and returns the lower half of the out coming rectangle
* CalculateOutcomingHalf():Rectangle – Calculate sand returns the out coming half of the out coming rectangle
* CalculateUpperHalfPoint(): Point – Calculates and returns the point that will be used for the pipeline connection of the upper half.
* CalculateLowerHalfPoint():Point – Calculates and returns the point that will be used for the pipeline connection of the lower half.
* CalculateOutcomingHalPointf(): Point – Calculates and returns the point that will be used for the pipeline connection of the out coming half.
* GetHalfOfComponent(Point p):Rectangle – returns either the lower or the upper half, based on the passed parameter.
* AdjustPipelineValues():void – Part of the IFlowHandler interface. Called when the Pipeline value is raise and updates all the values of the connected pipelines.

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

*Fields:*

* pipelineLocation:Point

*Properties:*

* OutcomePipeline: Pipeline- the pipeline that comes from that pump.
* Flow: double– the current flow of the pump
* Capacity:double – the capacity of the pump

*Constructor:*

Pump(int locx, int locy):base(locX,locY)

*Methods:*

* CalculatePipelineLocation():Point – returns a point that will be used for connecting pipelines.
* SetFlow(int newFlow):void – sets the flow of the pump if it does not exceed its capacity.
* GetFlow():String – returns a string with the flow and the capacity of the pump
* GetTextLocation():Point – returns a point that is used for drawing the information about the flow and capacity

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

*Fields:*

* pipelineLocation:Point

*Properties:*

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

*Constructor:*

* Sink(int locx, int locy):base(locX,locY)

*Methods:*

* CalculatePipelineLocation():Point – returns a point that will be used for connecting pipelines.
* GetFlow():String – returns a string with the flow of the sink
* GetTextLocation():Point – returns a point that is used for drawing the information about the flow

**Pipeline**

*Properties:*

* CurrentFlow:int – the current flow of the pipeline.
* MaxFlow:int- the max flow of the pipeline.
* InBetweenPoints:IList<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: Colour – returns the colour of the pipeline, accordingly to the current flow.
* StartPoint:Point
* EndPoint:Point
* GetPipelineRectangle:Rectangle

*Constructor:*

-Pipeline(Component startCom, Cpomponent endCom, Point stratCompLoc, Point endComp location, IList<Point> inBetweenpoints)

*Methods*:

* ChangeCurrentFlow(double newFlow):void – changes the current flow of a pipeline if it does not exceed its capacity
* ChangeMaxFlow(double newMaxFlow) :void – changes the max flow of the pipeline if it is bigger than the current flow
* ClearComponents():void - removes the pipeline from the components that it’s attached to
* GetFlow(): String – returns a string with the current and max flow of the pipeline.
* GetTextLocation():Point – returns a point that is used for drawing the information about the flow

**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.

Constructor:

* Network()
* Network(Serialization info, StreamingContext context)

*Methods:*

* SaveToFile(Network net, String path):static void – converts the file to binary and saves it in .xml format
* LoadFromFile(String path):Network – Deserialize a saved file and returns the loaded network.
* AddComponent(Component c):bool – adds a component to the list. Returns true if the component has been successfully added, otherwise false.
* RemoveComponent(Component c): void – removes a component from the list
* GetComponent(Point p):Component – returns the component localized on that point on the screen
* AddPipeline(Pipeline p):void – adds a pipeline to the list
* CreatePipeline(Component stratComp, Component endComp, Point stratComploc, Point endCompLoc, IList<Point>inBetweenpoints):Pipeline – Creates a pipeline between two components based on their location.
* RegisterPipeline(Component stratComp, Component endComp, Point stratComploc, Point endCompLoc, IList<Point>inBetweenpoints):void – Creates a pipeline using the private method CreatePipeline and adds the pipeline to the list.
* ValidatePipeline(Component stratComp, Component endComp, Point stratComploc, Point endCompLoc, IList<Point>inBetweenpoints):bool – validates whether the pipeline is eligible and should be created
* RemovePipeline(Component c):void – removes a pipeline from the list of pipelines and sets its starting and ending component to null
* RemovePipeline(Pipeline P): void – Removes a pipeline from the list and clears its starting and ending point.
* CreateComponent(ComponentType type, int locx, int locy):Component – creates a component in the specified location depending on the component type.
* GetObjectData(Serialization info, StreamingContext):void – Objects to be serialized and added to the serialization info

***ComponentType*** *enum –contains all the types of the components*

***IFlow****-Interface:*

*Methods:*

-GetFlow():String

- GetTextLocation():Point

***IFlowHandler*** *– Interface:*

*Methods:*

* + PipelineValueChanged: Action event
  + AdjustPipelineValues():void

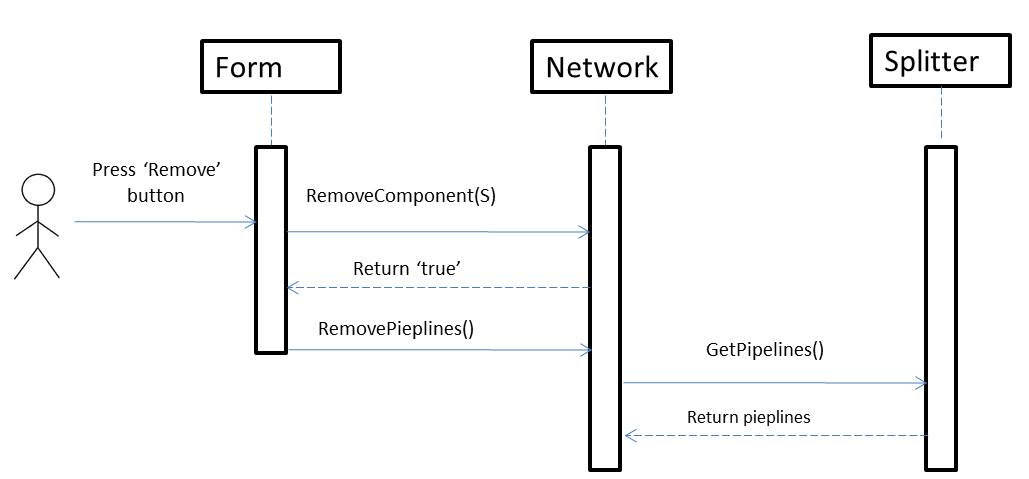
***ISplit -*** *Interface*

*Properties:*

* + UpeprHalf:Rectangle
  + LowerHalf:Rectangle

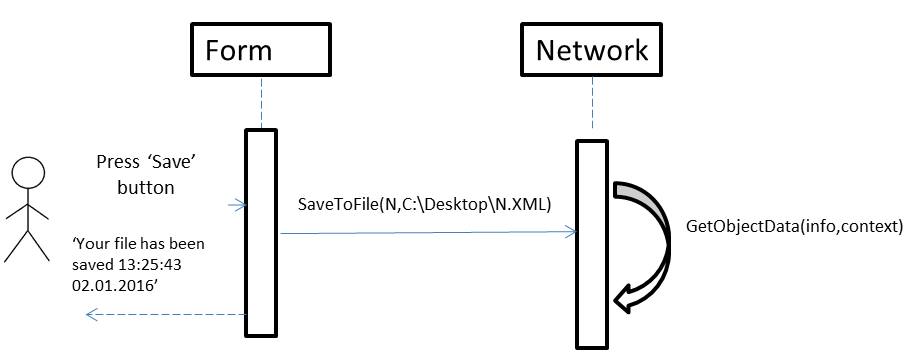
# Sequence Diagrams

### Remove component (splitter)



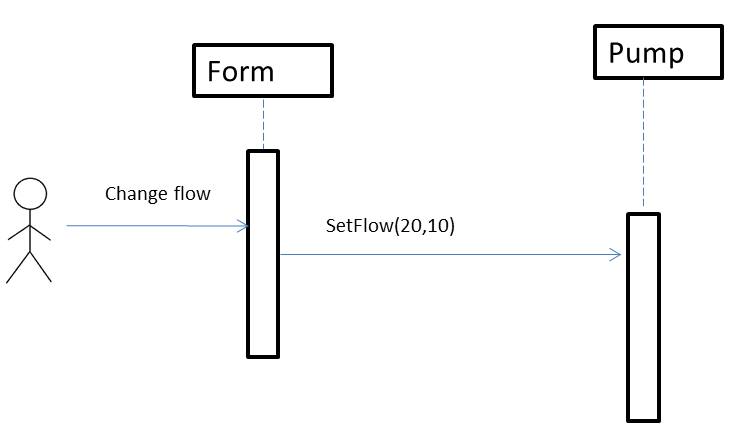
When the user wants to remove a component, he selects the component and presses the button Remove. Afterwards, the component is deleted from the system as well as the pipelines connected to it, if there are any

### Save file



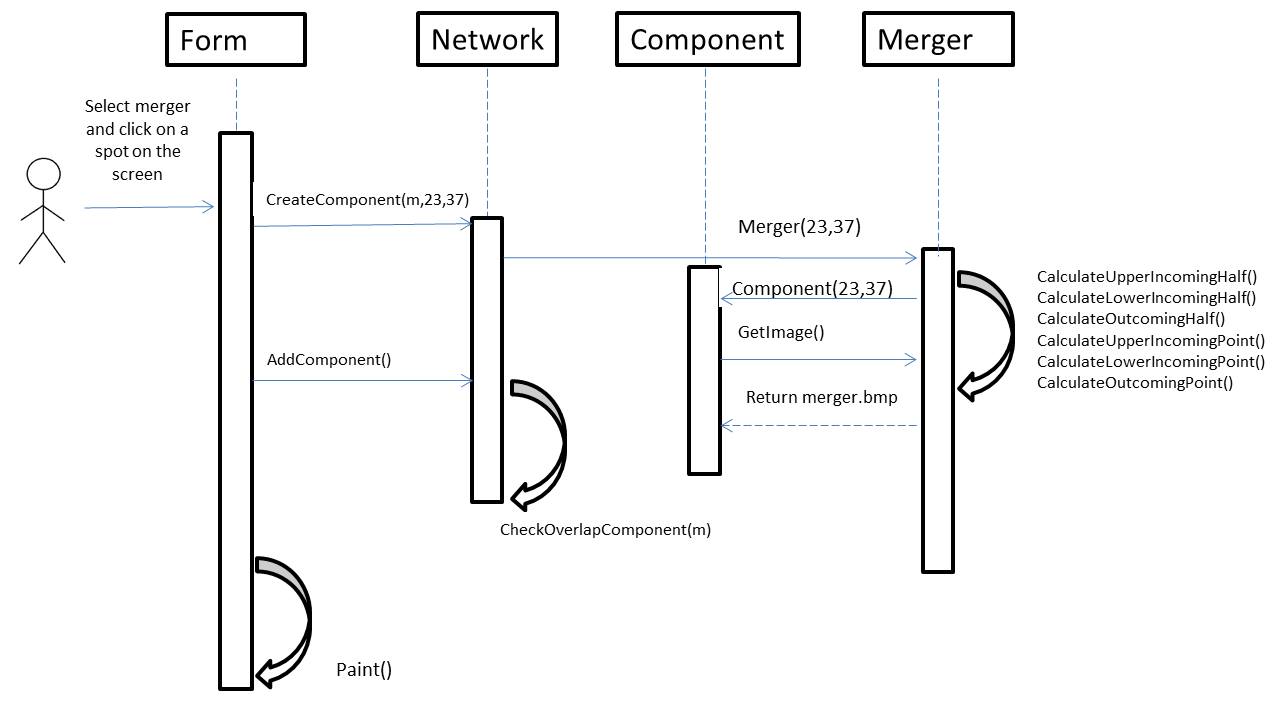
When saving a file, the system will convert the objects to binary. Afterwards, an informative message will be shown on the screen.

### 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.