The **classdiagram** package

Linus Sunde

July 20, 2013

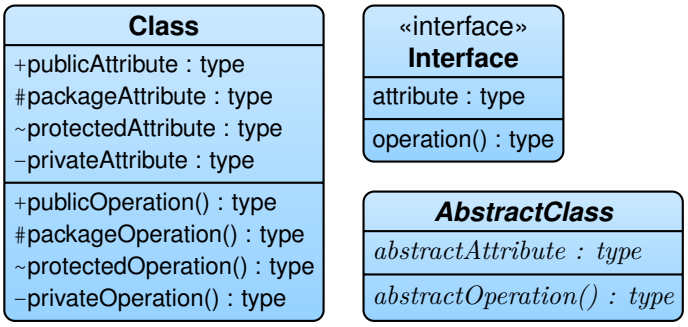
Abstract

The classdiagram package allows for creation of UML Class Diagrams. It uses PGF/TikZ to produces the vector graphics. The creation of the package was inspired by the pgf-umlcd package, which provides similar functionality. Many thanks to all the patient people at http://tex.stackexchange.com/ who took their time to answer my beginner questions. This package is a work in progress and might never be finished.

Contents

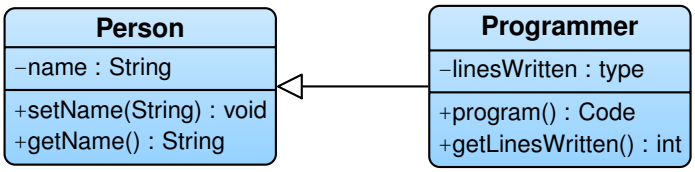
1 User Commands

1.1 **class** and **interface**



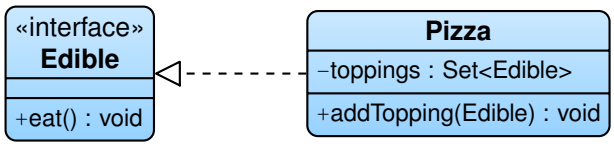

```
\begin{tikzpicture}

  \begin{class}{Class}  % Doesn't allow special characters
    \position{0, 0}
    \attribute[public]{publicAttribute : type}
    \attribute[protected]{packageAttribute : type}
    \attribute[package]{protectedAttribute : type}
    \attribute[private]{privateAttribute : type}
    \operation[public]{publicOperation() : type}
    \operation[protected]{packageOperation() : type}
    \operation[package]{protectedOperation() : type}
    \operation[private]{privateOperation() : type}
  \end{class}

  \begin{interface}{Interface}
    \position[right=0.5cm, anchor=north west]{Class.north east}
    \attribute{attribute : type}
    \operation{operation() : type}
  \end{interface}

  \begin{class}{AbstractClass}
    \position[right=0.5cm, anchor=south west]{Class.south east}
    \abstract
    \attribute*{abstractAttribute : type}
    \operation*{abstractOperation() : type}
  \end{class}

\end{tikzpicture}
```

1.2 \generalization (Class Inheritance)



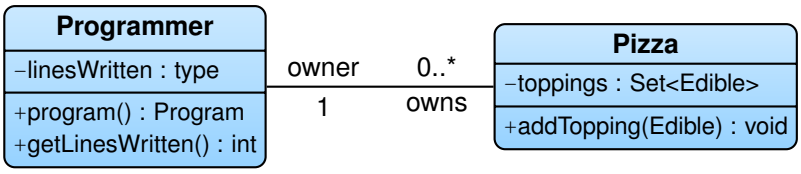

```
\begin{tikzpicture}

  \begin{class}{Person}
    \attribute[private]{name : String}
    \operation[public]{setName(String) : void}
    \operation[public]{getName() : String}
  \end{class}

  \begin{class}{Programmer}
    \position[right=2cm]{Person.east}
    \attribute[private]{linesWritten : type}
    \operation[public]{program() : Code}
    \operation[public]{getLinesWritten() : int}
  \end{class}

  \generalization{Programmer}{Person}

\end{tikzpicture}
```

1.3 \realization (Interface Implementation)



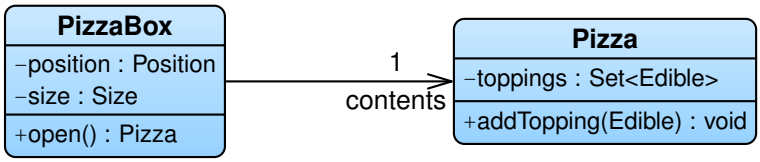

```
\begin{tikzpicture}

  \begin{interface}{Edible}
    \operation[public]{eat() : void}
  \end{interface}

  \begin{class}{Pizza}
    \position[right=2cm]{Edible.east}
    \attribute[private]{toppings : Set<Edible>}
    \operation[public]{addTopping(Edible) : void}
  \end{class}

  \realization{Pizza}{Edible}

\end{tikzpicture}
```

1.4 \association



```
\begin{tikzpicture}

  \begin{class}{Programmer}
    \attribute[private]{linesWritten : type}
    \operation[public]{program() : Program}
    \operation[public]{getLinesWritten() : int}
  \end{class}

  \begin{class}{Pizza}
    \position[right=3cm]{Programmer.east}
    \attribute[private]{toppings : Set<Edible>}
    \operation[public]{addTopping(Edible) : void}
  \end{class}

  \association{Programmer}[owns][0..*]{Pizza}[owner][1]

\end{tikzpicture}
```

1.5 \association[unidirectional]

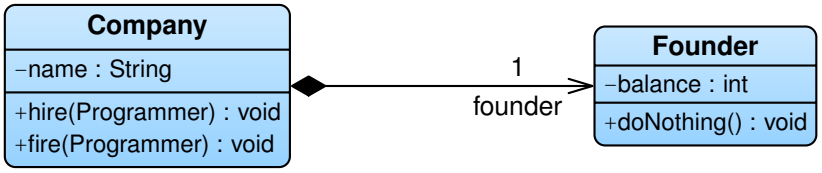



```
\begin{tikzpicture}

  \begin{class}{PizzaBox}
    \attribute[private]{position : Position}
    \attribute[private]{size : Size}
    \operation[public]{open() : Pizza}
  \end{class}

  \begin{class}{Pizza}
    \position[right=3cm]{PizzaBox.east}
    \attribute[private]{toppings : Set<Edible>}
    \operation[public]{addTopping(Edible) : void}
  \end{class}

  \association[unidirectional]{PizzaBox}[contents][1]{Pizza}

\end{tikzpicture}
```

1.6 \aggregation

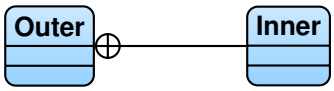
```
\begin{tikzpicture}

  \begin{class}{Company}
    \attribute[private]{name : String}
    \operation[public]{hire(Programmer) : void}
    \operation[public]{fire(Programmer) : void}
  \end{class}

  \begin{class}{Programmer}
    \position[right=4cm]{Company.east}
    \attribute[private]{linesWritten : type}
    \operation[public]{program() : Program}
    \operation[public]{getLinesWritten() : int}
  \end{class}

  \aggregation{Company}[employees][0..*]{Programmer}

\end{tikzpicture}
```

1.7 \composition




```
\begin{tikzpicture}

  \begin{class}{Company}
    \attribute[private]{name : String}
    \operation[public]{hire(Programmer) : void}
    \operation[public]{fire(Programmer) : void}
  \end{class}

  \begin{class}{Founder}
    \position[right=4cm]{Company.east}
    \attribute[private]{balance : int}
    \operation[public]{doNothing() : void}
  \end{class}

  \composition{Company}[founder][1]{Founder}

\end{tikzpicture}
```

1.8 \nested



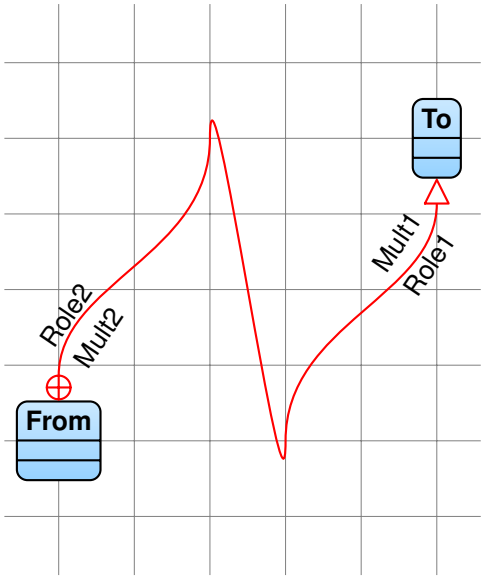

```
\begin{tikzpicture}

  \begin{class}{Outer}
  \end{class}

  \begin{class}{Inner}
    \position[right=2cm]{Outer.east}
  \end{class}

  \nested{Outer}{Inner}

\end{tikzpicture}
```

1.9 \relation

All of the above relations uses `\relation` behind the scenes.

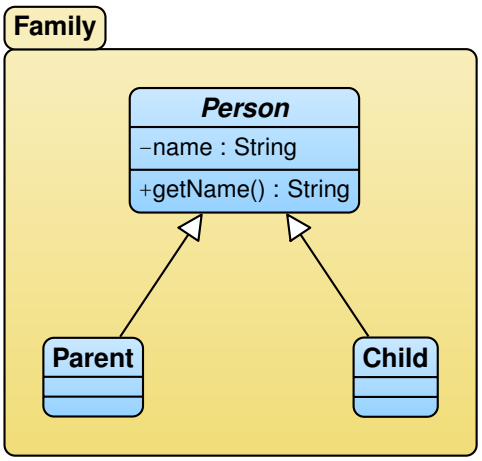
```
\relation[options]{from}[role][multiplicity]{to}[role][multiplicity]<via>
```

The `\association` example demonstrated what all the arguments except `[options]` and `<via>` do. First, `[options]` specifiy TikZ options to apply to the relation's "arrow". Secondly, `<via>` allows for specification of a list of intermediary points, via which the "arrow" will pass. All of these arguments are available for all relation types, but they do not always make sense to use. The specification of this function is subject to change. It should for example be possible to provide TikZ options for each coordinate in `<via>`.




```
\begin{tikzpicture}[show background grid]

  \begin{class}{From}
    \position{0, -2}
  \end{class}

  \begin{class}{To}
    \position{5, 2}
  \end{class}

  \relation[red, out = 90, in = -90, ox-i>]{From}[Role1][Mult1]{To}[Role2][Mult2]<(2,2),(
      3,-2)>

\end{tikzpicture}
```

1.10 \package



```
\begin{tikzpicture}

  \begin{class}{Person}
    \abstract
    \attribute[private]{name : String}
    \operation[public]{getName() : String}
  \end{class}

  \begin{class}{Parent}
    \position{-2, -3}
  \end{class}

  \begin{class}{Child}
    \position{2, -3}
  \end{class}

  \generalization{Parent}{Person}
  \generalization{Child}{Person}
  \package{Family}{(Person)(Parent)(Child)}

\end{tikzpicture}
```

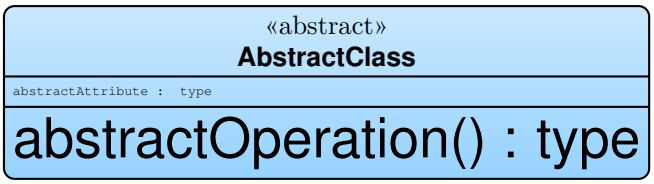
2 Configuration

2.1 Text

The `classdiagram` package defines a number of macros that are used to generate the recurring texts. These can be redefined to change the text and/or style.

`\textAbstractClass`	Defines the stereotype used for abstract classes:
`\textInterface`	Defines the stereotype used for interfaces: «interface»

`\styleClass`	sffamily bfseries	**Example**
`\styleAttribute`	sffamily small	Example
`\styleOperation`	sffamily small	Example
`\stylePackage`	sffamily bfseries	**Example**
`\styleAbstractClass`	itshape	*Example*
`\styleAbstractAttribute`	itshape	*Example*
`\styleAbstractOperation`	itshape	*Example*

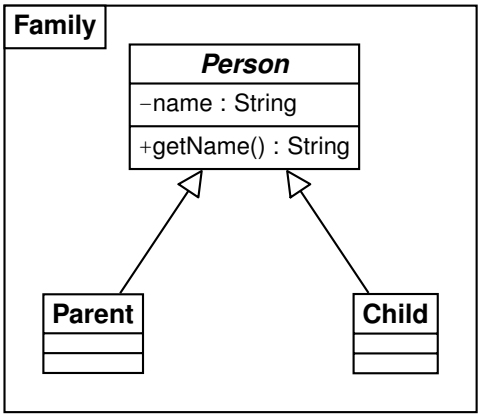



```
\begin{tikzpicture}

  \renewcommand{\textAbstractClass}{<<abstract>> \\}
  \renewcommand{\styleAbstractClass}{\sffamily \bfseries}
  \renewcommand{\styleAbstractAttribute}{\tiny\ttfamily}
  \renewcommand{\styleAbstractOperation}{\huge\sffamily}

  \begin{class}{AbstractClass}
    \position[right=0.5cm, anchor=south west]{Class.south east}
    \abstract
    \attribute*{abstractAttribute : type}
    \operation*{abstractOperation() : type}
  \end{class}

\end{tikzpicture}
```

2.2 Styles

The `classdiagram` package defines a number of TikZ styles that are used to generate the class diagrams, and which can be overridden to change the look and feel.

cd relation
cd relation text
cd relation role start
cd relation role end
cd relation multiplicity start
cd relation multiplicity end
cd nested
cd association
cd unidirectional
cd generalization
cd realization
cd aggregation
cd composition
cd package
cd package box
cd package label



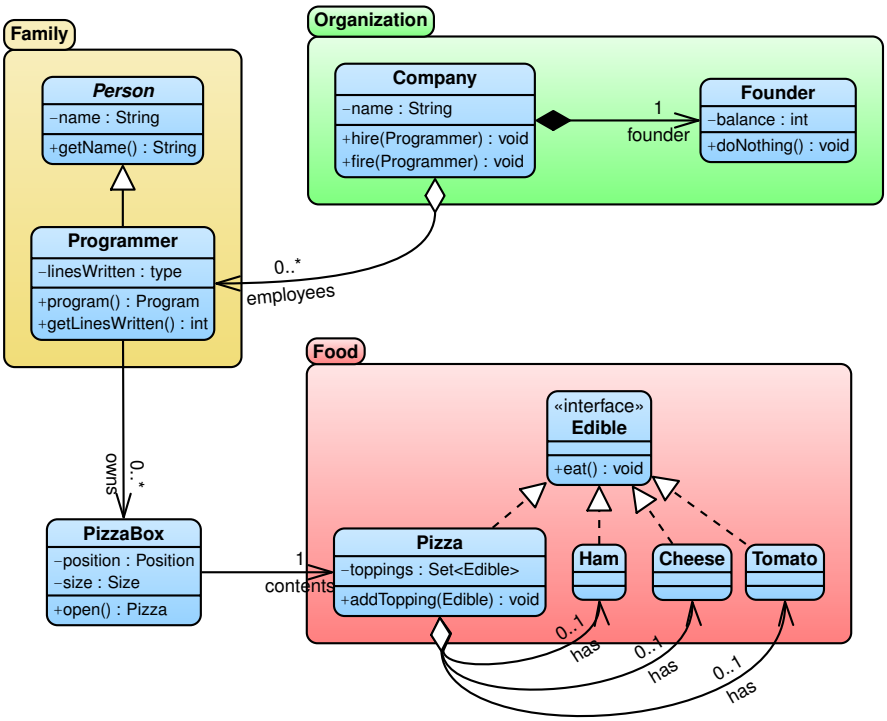

```
\begin{tikzpicture}[
  cd common/.append style={
    sharp corners, bottom color=white, top color=white},
  cd package label/.append style={
    sharp corners, fill=white, anchor=north west, outer ysep=0.4pt},
  cd package box/.append style=
    {sharp corners, bottom color=white, top color=white}]

  \begin{class}{Person}
    \abstract
    \attribute[private]{name : String}
    \operation[public]{getName() : String}
  \end{class}

  \begin{class}{Parent}
    \position{-2, -3}
  \end{class}

  \begin{class}{Child}
    \position{2, -3}
  \end{class}

  \generalization{Parent}{Person}
  \generalization{Child}{Person}
  \package{Family}{(Person)(Parent)(Child)}

\end{tikzpicture}
```

2.3 Arrows

The `classdiagram` package defines a number of TikZ arrowheads.

`-a>`
`-i>`
`ox-`
`<>-`
`<*>-`

3 Example


Family
Person
−name : String
+getName() : String
Programmer
−linesWritten : type
+program() : Program
+getLinesWritten() : int
Organization
Company
−name : String
+hire(Programmer) : void
+fire(Programmer) : void
1
founder
Founder
−balance : int
+doNothing() : void
0..*
employees
owns
0..*
PizzaBox
−position : Position
−size : Size
+open() : Pizza
1
contents
Food
«interface»
Edible
+eat() : void
Pizza
−toppings : Set<Edible>
+addTopping(Edible) : void
Ham
Cheese
Tomato
0..1
has
0..1
has
0..1
has

```latex
\begin{tikzpicture}[scale=0.7, every node/.style={transform shape}]

  \begin{class}{Person}
    \abstract
    \attribute[private]{name : String}
    \operation[public]{getName() : String}
  \end{class}

  \begin{class}{Company}
    \position[right=4]{Person}
    \attribute[private]{name : String}
    \operation[public]{hire(Programmer) : void}
    \operation[public]{fire(Programmer) : void}
  \end{class}

  \begin{class}{Founder}
    \position[right=5]{Company}
    \attribute[private]{balance : int}
    \operation[public]{doNothing() : void}
  \end{class}

  \begin{class}{Programmer}
    \position[below=2]{Person}
    \attribute[private]{linesWritten : type}
    \operation[public]{program() : Program}
    \operation[public]{getLinesWritten() : int}
  \end{class}

  \begin{interface}{Edible}
    \position{9, -6}
    \operation[public]{eat() : void}
  \end{interface}

  \begin{class}{Ham}
    \position[below=2]{Edible}
  \end{class}

  \begin{class}{Cheese}
    \position[right=1]{Ham}
  \end{class}

  \begin{class}{Tomato}
    \position[right=1]{Cheese}
  \end{class}

  \begin{class}{Pizza}
    \position[left=1]{Ham}
    \attribute[private]{toppings : Set<Edible>}
    \operation[public]{addTopping(Edible) : void}
  \end{class}

  \begin{class}{PizzaBox}
    \position[left=4.5]{Pizza}
    \attribute[private]{position : Position}
    \attribute[private]{size : Size}
    \operation[public]{open() : Pizza}
  \end{class}

  \aggregation[bend right=90]{Pizza}[has][0..1]{Ham}
  \aggregation[bend right=90]{Pizza}[has][0..1]{Cheese}
  \aggregation[bend right=90]{Pizza}[has][0..1]{Tomato}
  \realization{Ham}{Edible}
  \realization{Cheese}{Edible}
  \realization{Tomato}{Edible}
  \realization{Pizza}{Edible}
  \association[unidirectional]{PizzaBox}[contents][1]{Pizza}
  \association[unidirectional]{Programmer}[owns][0..*]{PizzaBox}
  \aggregation[out=-90, in=0]{Company}[employees][0..*]{Programmer}
  \composition{Company}[founder][1]{Founder}
  \generalization{Programmer}{Person}
  \package{Family}{(Person)(Programmer)}
  \package[bottom color=green!50, top color=green!10]{Organization}{(Company)(Founder)}
  \package[bottom color=red!50, top color=red!10]{Food}{(Pizza)(Ham)(Cheese)(Tomato)(
      Edible)}

\end{tikzpicture}
```