

Drawing UML Class Diagram by using `pgf-umlcd`

Yuan Xu

January 31, 2012 (v0.2.1)

Abstract

`pgf-umlcd` is a LaTeX package for drawing UML Class Diagrams. As stated by its name, it is based on a very popular graphic package PGF/TikZ. This document presents the usage of `pgf-umlcd` and collects some UML class diagrams as examples. `pgf-umlcd` can be downloaded from <http://code.google.com/p/pgf-umlcd/>.

Contents

1	Basics	1
1.1	Class with attributes and operations	1
1.1.1	Visibility of attributes and operations	2
1.1.2	Abstract class and interface	2
1.1.3	Object	2
1.1.4	Note	3
1.2	Inheritance and implement	3
1.2.1	Inheritance	3
1.2.2	Implement an interface	4
1.3	Association, Aggregation and Composition	4
1.3.1	Association	4
1.3.2	Unidirectional association	5
1.3.3	Aggregation	5
1.3.4	Composition	5
1.4	Package	6
2	Examples	6
2.1	Abstract Factory	6
3	Acknowledgements	7

1 Basics

1.1 Class with attributes and operations

Note: If you don't want to show empty parts in the diagrams, please use `simplified` option, e.g. `\usepackage[simplified]{pgf-umlcd}`.

ClassName
name : attribute type
name : attribute type = default value
name(parameter list) : type of value returned
<i>name(parameters list) : type of value returned</i>

```
\begin{tikzpicture}
  \begin{class}[text width=8cm]{ClassName}{0,0}
    \attribute{name : attribute type}
    \attribute{name : attribute type = default value}

    \operation{name(parameter list) : type of value
      returned}
    % virtual operation
    \operation[0]{name(parameters list) : type of
      value returned}
  \end{class}
\end{tikzpicture}
```

1.1.1 Visibility of attributes and operations

Class
+ Public
Protected
- Private
~ Package

BankAccount
+ owner : String
+ balance : Dollars
+ deposit(amount : Dollars)
+ withdrawal(amount : Dollars)
updateBalance(newBalance : Dollars)

```

\begin{tikzpicture}%[show background grid]
  \begin{class}[text width=7cm]{Class}{0,0}
    \attribute{+ Public}
    \attribute{\# Protected}
    \attribute{- Private}
    \attribute{\$ \sim$ Package}
  \end{class}

  \begin{class}[text width=7cm]{BankAccount}{0,-3}
    \attribute{+ owner : String}
    \attribute{+ balance : Dollars}

    \operation{+ deposit( amount : Dollars )}
    \operation{+ withdrawal( amount : Dollars )}
    \operation{\# updateBalance( newBalance : Dollars
    )}
  \end{class}
\end{tikzpicture}

```

1.1.2 Abstract class and interface

<<abstract>> BankAccount
owner : String
balance : Dollars = 0
deposit(amount : Dollars)
<i>withdrawal(amount : Dollars)</i>

<<interface>> Person
firstName : String
lastName : String

```

\begin{tikzpicture}
  \begin{abstractclass}[text width=5cm]{BankAccount
  }{0,0}
    \attribute{owner : String}
    \attribute{balance : Dollars = 0}

    \operation{deposit(amount : Dollars)}
    \operation[0]{withdrawal(amount : Dollars)}
  \end{abstractclass}
\end{tikzpicture}

```

```

\begin{tikzpicture}%[show background grid]
  \begin{interface}{Person}{0,0}
    \attribute{firstName : String}
    \attribute{lastName : String}
  \end{interface}
\end{tikzpicture}

```

1.1.3 Object

<u>Instance Name: Class Name</u>
attribute name = value

```

\begin{tikzpicture}
  \begin{object}[text width=6cm]{Instance Name}{0,0}
    \instanceOf{Class Name}
    \attribute{attribute name = value}
  \end{object}
\end{tikzpicture}

```

Note: Object with rounded corners and methods are used in German school for didactic reasons. You get the rounded corners with `\usepackage[school]{pgf-umlcd}`. If you need both in one document you can switch it with `\switchUmlcdSchool`

<u>Instance Name: Class Name</u>
attribute name = value

```

\begin{tikzpicture}
  \begin{object}[text width=6cm]{Instance Name}{0,0}
    \instanceOf{Class Name}
    \attribute{attribute name = value}
  \end{object}
\end{tikzpicture}

```

Thomas' account: BankAccount

owner = Thomas

balance = 100

deposit(amount : Dollars)

withdrawl(amount : Dollars)

```
\begin{tikzpicture}
  \begin{object}[text width=6cm]{Thomas' account}
    \instanceOf{BankAccount}
    \attribute{owner = Thomas}
    \attribute{balance = 100}

    \operation{deposit(amount : Dollars)}
    \operation[0]{withdrawl(amount : Dollars)}
  \end{object}
\end{tikzpicture}
```

1.1.4 Note

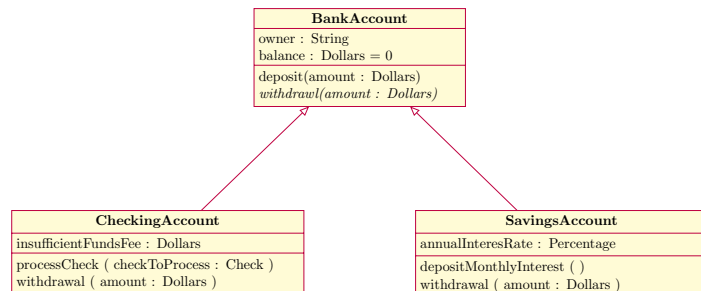
The `\umlnote` use the same syntax as tikz command `\node`, e.g. `\umlnote[style] (name)at (coordinate){text};`

This is a note.

```
\begin{tikzpicture}
  \umlnote (note) {This is a note.};
\end{tikzpicture}
```

1.2 Inheritance and implement

1.2.1 Inheritance



```
\begin{tikzpicture}
  \begin{class}[text width=5cm]{BankAccount}{0,0}
    \attribute{owner : String}
    \attribute{balance : Dollars = 0}

    \operation{deposit(amount : Dollars)}
    \operation[0]{withdrawl(amount : Dollars)}
  \end{class}

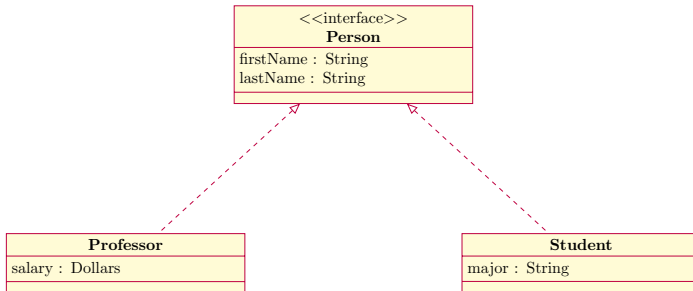
  \begin{class}[text width=7cm]{CheckingAccount}
    {-5,-5}
    \inherit{BankAccount}
    \attribute{insufficientFundsFee : Dollars}

    \operation{processCheck ( checkToProcess : Check )}
    \operation{withdrawal ( amount : Dollars )}
  \end{class}

  \begin{class}[text width=7cm]{SavingsAccount}{5,-5}
    \inherit{BankAccount}
    \attribute{annualInteresRate : Percentage}

    \operation{depositMonthlyInterest ( )}
    \operation{withdrawal ( amount : Dollars )}
  \end{class}
\end{tikzpicture}
```

1.2.2 Implement an interface



```

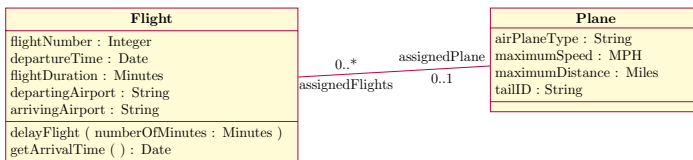
\begin{tikzpicture}%[show background grid]
\begin{interface}{Person}{0,0}
\attribute{firstName : String}
\attribute{lastName : String}
\end{interface}

\begin{class}{Professor}{-5,-5}
\implement{Person}
\attribute{salary : Dollars}
\end{class}

\begin{class}{Student}{5,-5}
\implement{Person}
\attribute{major : String}
\end{class}
\end{tikzpicture}
  
```

1.3 Association, Aggregation and Composition

1.3.1 Association



```

\begin{tikzpicture}
\begin{class}[text width=7cm]{Flight}{0,0}
\attribute{flightNumber : Integer}
\attribute{departureTime : Date}
\attribute{flightDuration : Minutes}
\attribute{departingAirport : String}
\attribute{arrivingAirport : String}

\operation{delayFlight ( numberOfMinutes :
Minutes )}
\operation{getArrivalTime ( ) : Date}
\end{class}

\begin{class}{Plane}{11,0}
\attribute{airPlaneType : String}
\attribute{maximumSpeed : MPH}
\attribute{maximumDistance : Miles}
\attribute{tailID : String}
\end{class}

\association{Plane}{assignedPlane}{0..1}{Flight
}{0..*}{assignedFlights}
\end{tikzpicture}
  
```

1.3.2 Unidirectional association



```

\begin{tikzpicture}
% \draw[help lines] (-7,-6) grid (6,0);

\begin{class}[text width=6cm]{
  OverdrawnAccountsReport}{0,0}
  \attribute{generatedOn : Date}

  \operation{refresh ( )}
\end{class}

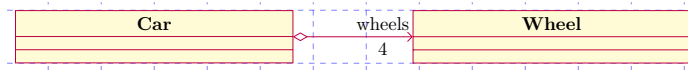
\begin{class}{BankAccount}{12,0}
  \attribute{owner : String}
  \attribute{balance : Dollars}

  \operation{deposit(amount : Dollars)}
  \operation[0]{withdrawl(amount : Dollars)}
\end{class}

\unidirectionalAssociation{OverdrawnAccountsReport
  }{overdrawnAccounts}{0..*}{BankAccount}

\end{tikzpicture}
  
```

1.3.3 Aggregation



```

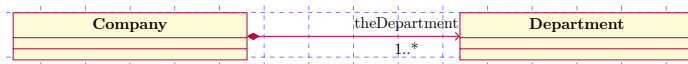
\begin{tikzpicture}[show background grid]
\begin{class}{Car}{0,0}
\end{class}

\begin{class}{Wheel}{7.5,0}
\end{class}

\aggregation{Car}{wheels}{4}{Wheel}

\end{tikzpicture}
  
```

1.3.4 Composition



```

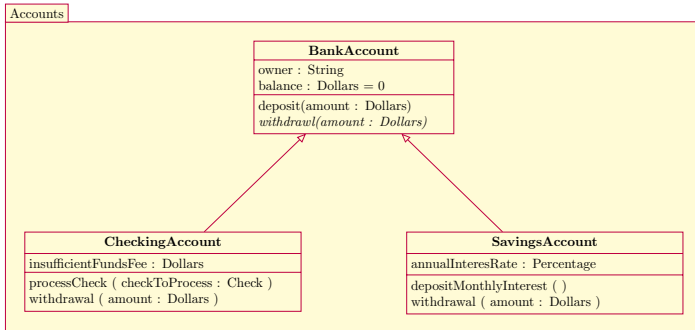
\begin{tikzpicture}[show background grid]
\begin{class}{Company}{0,0}
\end{class}

\begin{class}{Department}{10,0}
\end{class}

\composition{Company}{theDepartment}{1..*}{
  Department}

\end{tikzpicture}
  
```

1.4 Package



```

\begin{tikzpicture}
  \begin{package}{Accounts}
    \begin{class}[text width=5cm]{BankAccount}{0,0}
      \attribute{owner : String}
      \attribute{balance : Dollars = 0}

      \operation{deposit(amount : Dollars)}
      \operation[0]{withdrawal(amount : Dollars)}
    \end{class}

    \begin{class}[text width=7cm]{CheckingAccount}
      {-5,-5}
      \inherit{BankAccount}
      \attribute{insufficientFundsFee : Dollars}

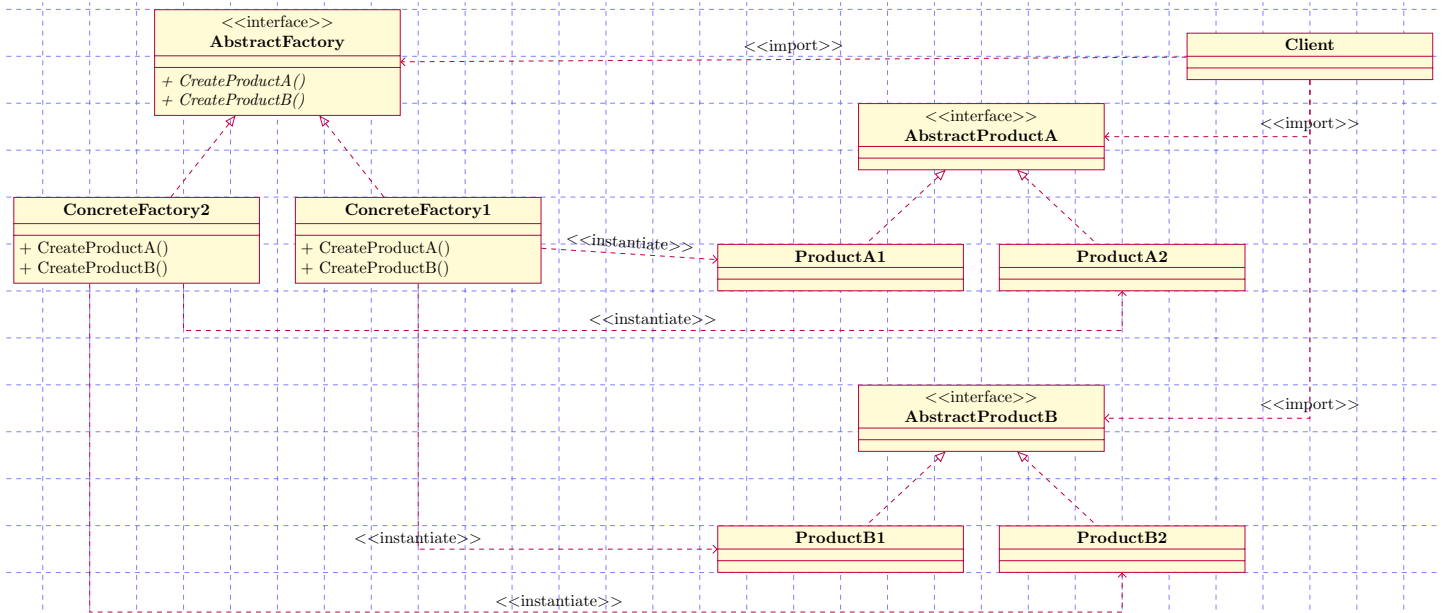
      \operation{processCheck ( checkToProcess :
        Check )}
      \operation{withdrawal ( amount : Dollars )}
    \end{class}

    \begin{class}[text width=7cm]{SavingsAccount}
      {5,-5}
      \inherit{BankAccount}
      \attribute{annualInterestRate : Percentage}

      \operation{depositMonthlyInterest ( )}
      \operation{withdrawal ( amount : Dollars )}
    \end{class}
  \end{package}
\end{tikzpicture}
  
```

2 Examples

2.1 Abstract Factory



```

\begin{tikzpicture}[show background grid]
  \begin{interface}{AbstractFactory}{0,0}
    \operation[0]{+ CreateProductA()}
    \operation[0]{+ CreateProductB()}
  \end{interface}

  \begin{class}{ConcreteFactory2}{-3,-4}
    \implement{AbstractFactory}
  \end{class}
  
```

```

\operation{+ CreateProductA()}
\operation{+ CreateProductB()}
\end{class}

\begin{class}{ConcreteFactory1}{3,-4}
\implement{AbstractFactory}
\operation{+ CreateProductA()}
\operation{+ CreateProductB()}
\end{class}

\begin{interface}{AbstractProductA}{15,-2}
\end{interface}

\begin{class}{ProductA1}{12,-5}
\implement{AbstractProductA}
\end{class}

\begin{class}{ProductA2}{18,-5}
\implement{AbstractProductA}
\end{class}

\draw[umlcd style dashed line,->] (ConcreteFactory1) --node[above,
sloped, black]{$<<$instantiate$>>$} (ProductA1);

\draw[umlcd style dashed line,->] (ConcreteFactory2.south) ++
(1,0) -- ++(0,-1) -- node[above, sloped,
black]{$<<$instantiate$>>$} ++(20,0) -| (ProductA2);

\begin{interface}{AbstractProductB}{15,-8}
\end{interface}

\begin{class}{ProductB1}{12,-11}
\implement{AbstractProductB}
\end{class}

\begin{class}{ProductB2}{18,-11}
\implement{AbstractProductB}
\end{class}

\draw[umlcd style dashed line,->] (ConcreteFactory1) |-node[above,
sloped, black]{$<<$instantiate$>>$} (ProductB1);

\draw[umlcd style dashed line,->] (ConcreteFactory2.south) ++
(-1,0) -- ++(0,-7) -- node[above, sloped,
black]{$<<$instantiate$>>$} ++(20,0) -| (ProductB2);

\begin{class}{Client}{22,-0.5}
\end{class}

\draw[umlcd style dashed line,->] (Client) --node[above, sloped,
black]{$<<$import$>>$} (AbstractFactory);

\draw[umlcd style dashed line,->] (Client) |-node[above, sloped,
black]{$<<$import$>>$} (AbstractProductA);

\draw[umlcd style dashed line,->] (Client) |-node[above, sloped,
black]{$<<$import$>>$} (AbstractProductB);
\end{tikzpicture}

```

3 Acknowledgements

Many people contributed to pgf-umlcd by reporting problems, suggesting various improvements or submitting code. Here is a list of these people: [Martin Quinson](#), and [Johannes Pieper](#).