UML Cheatsheet

UML diagrams are used for communication which convey meaningful parts of your application. Include the data which will help someone better understand your code, not everything must be included

Representing Classes:

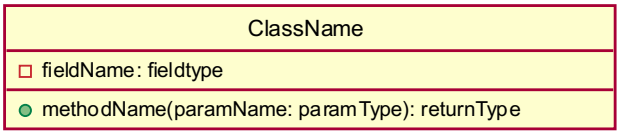
The technique for representing in UML Java classes, fields, and Java methods is:

Fields:
fieldName: Type

Methods:
methodName(paramName1: Type, paramName2: Type): methodReturnType

Below is a general template for representing classes, and a small representation of the String class. If you're representing an interface, put <<interface>> above the class name, for an abstract class, put the name in *italics*.

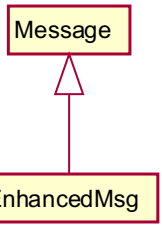
Template:



PlantUML:

```
class ClassName {
   -fieldName: fieldtype
   +methodName(paramName: paramType): returnType
}
```

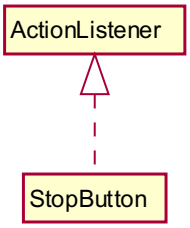
Arrows:



Inheritance
is-a
extends a class

EnhancedMsg is-a Message

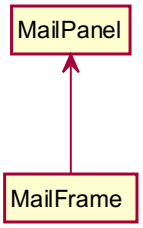
PlantUML:
`EnhancedMsg -|> Message`



Interface Implementation
is-a
implements an interface

StopButton is-a ActionListener

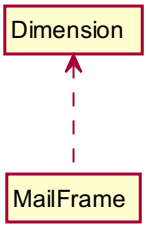
PlantUML:
```
StopButton ..|>
   ActionListener
```



Association
has-a field

MailFrame has-a MailPanel field

PlantUML:
`MailFrame -> MailPanel`

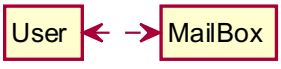


Dependency
depends-on
but is not a field

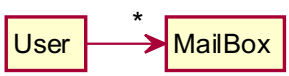
MailFrame depends-on Dimension

PlantUML:
`MailFrame ..> Dimension`

Two-Way Association
User has a MailBox field AND MailBox has a User field
PlantUML: `User <-> MailBox`



Two-Way Dependency
User depends on MailBox AND MailBox depends on User
(A MailBox method takes a User as a parameter, or vice versa)
PlantUML: `User <..> MailBox`



Cardinality
How many items are in a relationship: '*' = 0..infinity
Say: "One User has many MailBoxes"
PlantUML: `User ->"*" MailBox`