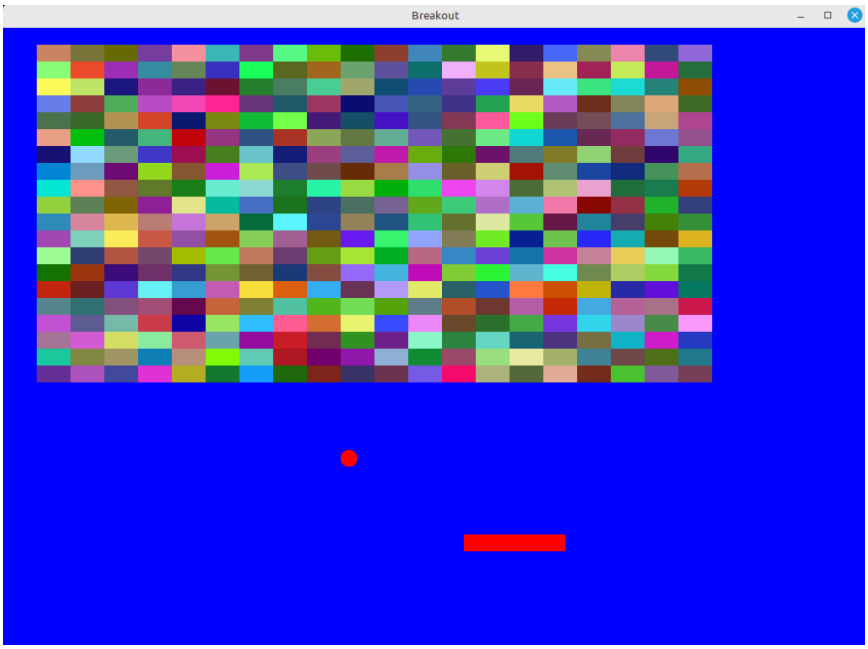
usepackagedejavu
renewcommand*
familydefault
ttdefault [[file:dog.jpg]]
$parbox5cmnormal font This text should be displayed to the right of the image above. Ideally, this would work$

Contents

1 Motivating example - make a bouncing ball game in Bloc



1.1 Have keyboard input and actions take place when press left / right –

1.2 Build a matrix of coloured bricks ?

1.3 Have bouncing ball ?

1.4 Open a window

1.5 Close a window

1.6 Get size of window

1.7 have a title - game score

1.8 play sound when

1.9 Collision detection ?

1.10 can we add event listening to it ?

1.11 can we do move over event ? we have seen that before

2 LATEST

Currently writing Breakout
terryc321.github.com/Breakout

3 Kill inspectors

```
SystemWindow allInstances
  select: [ :window |
    (window model isKindOf: StInspector) or: [
      (window model isKindOf: GTInspector) or: [
        window model isKindOf: Inspector ] ] ]
  thenDo: [ :window | window delete ]
```

4 Keyboard event handling

see lower down - find keylogger

5 Mouse event handling

mouse over mouse enter mouse leave click event double click event drag drop maybe ?

6 Spec

7 Spec2

8 GT World

9 Toplo

10 Roassal

11 Coypu

12 TinyPaint

13 Alexandrie

This is low level stuff

```
FIXME Does not work !!!
Metacello new
    baseline: 'SpartaAlexandrie';
    repository: 'github://pharo-graphics/Sparta:dev/src';
    load.
```

14 Bloc

Bloc is a low-level UI infrastructure & framework for Pharo.
`https://github.com/pharo-graphics/Bloc`

14.1 Bloc baseline

The baseline for bloc is

```
Metacello new
        baseline: 'Bloc';
        repository: 'github://pharo-graphics/Bloc:master/src';
        load
```

14.2 Bloc spec

The spec for bloc is
    How do we use spec again ?

```
spec baseline: 'Bloc' with: [ spec repository: 'github://pharo-graphics/Bloc:v2.5.0/src
```

14.2.1 Bloc tutorial

This a memory card game written in Bloc , whats most useful is it comes
with some ready to use examples

```
Metacello new
          baseline: 'BlocMemoryTutorial';
          repository: 'github://pharo-graphics/Bloc-Memory-Tutorial/src';
          load
```

```
"MG Memory Game "
MGGame withEmoji .
MGGameElement openWithNumber .
```

14.2.2 KeyLogger

load BlocMemoryTutorial

- BlKeyCombinationExamplesTest

15 Bloc graphical elements

16 Rectangles

Lets start with a rectangle
    Its easiest to have a filled rectangle

```
demoRectangle
<demo>
```

```
<sampleInstance>
"Create an empty rectangle"
|rectangle|
rectangle := BlElement new
        background: Color red ;
        position: 200 @ 100 ;
        size: 150 @ 150;
        yourself.
^ rectangle.
```

16.1 And now rotated 45 degrees , this is rotated by top left corner

```
demoRectangle2
<demo>
<sampleInstance>
"Create an empty rectangle"
|rectangle|
rectangle := BlElement new
        background: Color black ;
        position: 200 @ 100 ;
        size: 150 @ 150;
        transformDo: [ :t | t rotateBy: 45 ] ;
        yourself.
^ rectangle.
```

16.2 solid rotated 45 already at top left

expect some of the rectangle to be clipped as some of it will be off screen

```
demoRectangle4
<demo>
<sampleInstance>
"Create an empty rectangle"
|rectangle|
rectangle := BlElement new
        background: Color black ;
        position: 0 @ 0 ;
        size: 150 @ 150;
        transformDo: [ :t | t rotateBy: 45 ] ;
        yourself.
```

```
^ rectangle.
```

17 BlAffineTransformationOrigin

BlAffineTransformationTopLeftOrigin ? BlAffineTransformationLeftCentre-Origin ? BlRotationTransformation ?

17.1 A hollow rectangle

```
demoRectangle5
<demo>
<sampleInstance>
"Create an empty rectangle"
|rectangle|
rectangle := BlElement new
        border: (BlBorder paint: Color black width: 2);
        position: 0 @ 0 ;
        size: 150 @ 150;
        transformDo: [ :t |
            t origin: (BlAffineTransformationTopLeftOrigin new);
            rotateBy: 45 ] ;
        yourself.
^ rectangle.
```

17.2 rotation about centre

```
demoRectangle3
<demo>
<sampleInstance>
"Create an empty rectangle"
|rectangle|
rectangle := BlElement new
        border: (BlBorder paint: Color black width: 2);
        position: 200 @ 100 ;
        size: 150 @ 150;
        transformDo: [ :t | t rotateBy: 45 ] ;
        yourself.

rectangle transformDo: [ :t |
    t origin: (BlAffineTransformationCenterOrigin new);
```

```
    rotateBy: 45 ].
^ rectangle.
```

<sampleInstance> pragma makes Pharo aware this is a graphical element and can be displayed. This lets Pharo place a green play button on system browser to mean click this and see it on screen.

<demo> pragma makes Pharo aware this is a demo -?

This is a filled rectangle 150,150 in size

```
demoRectangle2
<sampleInstance>
<demo>
"Create a red rectangle"
|rectangle|
rectangle := BlElement new
        background: Color red;  "putting background to a color makes it filled"
        size: 150 @ 150;
        yourself.
^ rectangle.
```

18 Bloc tutorial

This a memory card game written in Bloc , not sure if it makes sense.

```
Metacello new
          baseline: 'BlocMemoryTutorial';
          repository: 'github://pharo-graphics/Bloc-Memory-Tutorial/src';
          load

MGGame withEmoji .

MGGameElement openWithNumber .
```

19 Bloc graphics

`https://github.com/pharo-graphics/Bloc?tab=readme-ov-file`

Pharo 14 load this to start using Bloc

```
Metacello new
        baseline: 'Bloc';
```

```
        repository: 'github://pharo-graphics/Bloc:master/src';
        load
```

the baseline for use with projects

```
spec baseline: 'Bloc' with: [ spec repository: 'github://pharo-graphics/Bloc:v2.5.0/src
```

19.1 open a window BlSpace

```
aSpace := BlSpace new.
aSpace show.

"Edit the space's properties, like title and size"
aSpace title: 'Bloc basics'.
aSpace extent: 800 @ 600.
```

19.2 Lets make a rectangle !

```
  aSpace := BlSpace new.
aSpace show.

"Edit the space's properties, like title and size"
aSpace title: 'Bloc basics'.
aSpace extent: 800 @ 600.


"Create a red rectangle"
rectangle := BlElement new
        background: Color red;
        size: 150 @ 150;
        yourself.

"Add it to the space"
aSpace root addChild: rectangle.

"Update its properties"
rectangle
        background: Color lightBlue;
        position: 100 @ 100;
        border: (BlBorder paint: Color blue width: 10).
```

```
"Update its properties"
rectangle
        background: Color black;
        position: 400 @ 100;
        border: (BlBorder paint: Color red width: 5).

"remove it from the space"
"aSpace root removeChild: rectangle."
```

19.3 Lets make a circle

```
circle := BlElement new
        background: Color blue;
        geometry: BlCircleGeometry new;
        size: 80 @ 80;
        yourself.
rectangle addChild: circle.
```

19.4 Lets make the circle bigger and play with clipping

```
circle size: 300@300 .
rectangle clipChildren: false.
rectangle clipChildren: true.
```

19.5 we can resize the circle

```
circle transformDo: [ :builder | builder scaleBy: 1.2 ].
```

19.6 Animate the rectangle

```
"Animate opacity"
rectangle addAnimation: (BlOpacityAnimation new opacity: 0.5).

"Animate transformations"
fallAnimation := (BlTransformAnimation translate: 0 @ 200) absolute.
rectangle addAnimation: fallAnimation.
climbAnimation := (BlTransformAnimation translate: 0 @ 0) absolute.
rectangle addAnimation: climbAnimation.

"Create a sequence of animations"
```

```
animationSequence := BlSequentialAnimation withAll: { fallAnimation. climbAnimation }.
animationSequence beInfinite.
rectangle addAnimation: animationSequence
```

19.7 handle some events

```
  "Change color on click"
rectangle addEventHandlerOn: BlClickEvent do: [ :event | event target background: Color

"Animate on hover"
rectangle
        addEventHandlerOn: BlMouseEnterEvent
        do: [ :event | event target addAnimation: (BlOpacityAnimation new opacity: 0.2)
        addEventHandlerOn: BlMouseLeaveEvent
        do: [ :event | event target addAnimation: (BlOpacityAnimation new opacity: 1.0)
```

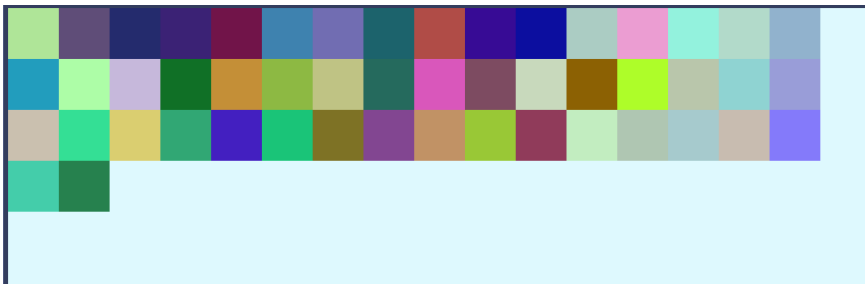
20 Bloc-Examples

Pharo playing with live objects `https://av.tib.eu/media/50551`

This package defines the examples for Bloc

20.1 BlMorphicHostExamples

This window is inside the smalltalk window , unlike the BlSpace example above .



```
BlMorphicHostExamples new squares .
```

How do i take a selected region screenshot in linux ? flameshot wow.

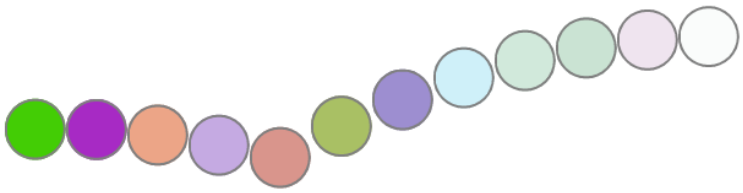
21 BlAnimationExamplesTest

```
BlAnimationExamplesTest new ballsAnim.
BlAnimationExamplesTest new bouncingText.
BlAnimationExamplesTest new sequential.
```

21.1 ballsAnim

when we run this it does nothing ?

We get a green triangle , we can play the animation .an iconicButtonMorph.



two pragmas

```
ballsAnim
"<sampleInstance>"
"<demo>"
    | elements animations |
        animations := OrderedCollection new.
        elements := OrderedCollection new.

        1 to: 12 do: [ :i |
                | anElement bounceAnimation colorizeAnimation |
                anElement :=
                        BlEllipseGeometry new asElement
                              background: Color white;
                              position: (i * 10) @ 0;
                              extent: 50 @ 50;
```

```
                        yourself.

                bounceAnimation :=
                        BlTransformAnimation new
                                target: anElement;
                                transformDo: [ :aBuilder |
                                        aBuilder translateBy:
                                                        0 @ 100 ];
                                delay: 100 milliSeconds * i;
                                duration: 2 seconds;
                                easing: BlEasing bounceOut;
                                yourself.

                colorizeAnimation :=
                        BlColorTransition new
                                from: Color white;
                                to: Color random;
                                delay: 100 milliSeconds * i;
                                duration: 1 second;
                                onStepDo: [ :c | anElement background: c ];
                                yourself.

                animations addAll: { bounceAnimation. colorizeAnimation }.
                elements add: anElement ].

        ^ self newFrameContainer
                addChildren: elements;
                addAnimation: (BlParallelAnimation withAll: animations);
                yourself
```

newFrameContainer method is

```
newFrameContainer

        ^ BlElement new
                  layout: BlLinearLayout horizontal alignCenter;
                  constraintsDo: [ :c |
                          c horizontal matchParent.
                          c vertical matchParent ];
                  clipChildren: false;
                  yourself
```

21.2 Lets draw a line

This works and draws a red line

```
| space lineElement |

"Create a BlElement with BlLineGeometry"
lineElement := BlElement new
    geometry: (BlLineGeometry from: 50@50 to: 200@200);
    border: (BlBorder paint: Color green width: 20);
    yourself.

"Set up the space"
space := BlSpace new.
    space root addChild: lineElement;
    extent: 400@300;
    yourself.

space show.
```

22 Polygons

22.1 Polygon no fill

some weird looking polygon

```
  | space polygonElement vertices |

"Define the vertices for a pentagon"
vertices := {
    100@50.  "Top"
    150@100. "Right-top"
    130@150. "Right-bottom"
    70@150.  "Left-bottom"
    50@100   "Left-top"
}.

"Create a BlElement with BlPolygonGeometry"
polygonElement := BlElement new
    geometry: (BlPolygonGeometry vertices: vertices);
    border: (BlBorder paint: Color red width: 3);
```

```
    background: Color transparent; "Ensure no fill"
    yourself.

"Set up the space"
space := BlSpace new.
    space root addChild: polygonElement;
    extent: 400@300;
    yourself.

space show.
```

22.2 Polygon with fill

Defines a space [a window opens separately]

```
| space polygonElement vertices |

"Define the vertices for a pentagon"
vertices := {
    100@50.  "Top"
    150@100. "Right-top"
    130@150. "Right-bottom"
    70@150.  "Left-bottom"
    50@100   "Left-top"
}.

"Create a BlElement with BlPolygonGeometry and fill"
polygonElement := BlElement new
    geometry: (BlPolygonGeometry vertices: vertices);
    background: Color red; "Fill color"
    border: (BlBorder paint: Color black width: 2); "Optional outline"
    yourself.

"Set up the space"
space := BlSpace new.
    space root addChild: polygonElement;
    extent: 400@300;
    yourself.

space show.
```

22.3 Polygon with fill

just describes the polygon element itself , no space window yet we get a green icon we can click , pharo 14 dev will create a window for us and place filled polygon into it

```
demoLine
<sampleInstance>
<demo>
| polygonElement vertices |

"Define the vertices for a pentagon"
vertices := {
    100@50.  "Top"
    150@100. "Right-top"
    130@150. "Right-bottom"
    70@150.  "Left-bottom"
    50@100   "Left-top"
}.

"Create a BlElement with BlPolygonGeometry and fill"
polygonElement := BlElement new
    geometry: (BlPolygonGeometry vertices: vertices);
    background: Color red; "Fill color"
    border: (BlBorder paint: Color black width: 2); "Optional outline"
    yourself.

^ polygonElement.
```

22.4 Bezier curve

openInWindow method

```
| p0 p1 p2 p3 canvas elem |
p0 := 20@140.
p1 := 120@20.
p2 := 280@220.
p3 := 360@60.

elem := BlElement new
```

```
    size: 400@250;
    background: Color white;
    yourself.

elem onPaint: [ :c |
    c path
        moveTo: p0;
        bezierVia: p1 and: p2 to: p3;
        stroke: (Color black width: 3).

    "Control lines"
    c path
        moveTo: p0; lineTo: p1;
        stroke: (Color gray width: 1).
    c path
        moveTo: p3; lineTo: p2;
        stroke: (Color gray width: 1).

    "Control points"
    {p0. p1. p2. p3} do: [:pt |
        c fillRectangle: (pt extent: 6@6) color: Color red ] ].

elem openInWindow
```

23 Saving private ryan

24 Dynamic class definition at runtime

If we wish to be in Smalltalk tradition everything must be dynamic , imagine we had to code entirely new graphical user interface.

24.1 make a class

pharo - ok : squeak - fails

```
Smalltalk classInstaller
    make: [ :builder |
        builder
```

```
        superclass: Object;
        name: #ZZFooBar2;
        slots: #(cow milk);
        classSlots: #();
        sharedPools: '';
        package: 'ZZPackage' ].
```

24.2 add a method with

```
(Smalltalk at: #ZZFooBar) compile: 'hello10 ^ 11' classified: nil.
(Smalltalk at: #ZZFooBar) compile: 'hello20 ^ 22' classified: 'magic number3'.
```

24.3 create a class side method

the message class returns the metaclass of receiver, in this case ZZFooBar class

```
(Smalltalk at: #ZZFooBar) class compile: 'goodbye ^ ''bye bye''' classified: 'magic nu
```

24.4 overwrite existing method

if hello method exists then compiling a new definition will overwrite old one

- may be cases where do not want this to happen

```
(Smalltalk at: #ZZFooBar) compile: 'hello ^ 123' classified: 'magic number3'.
```

24.5 get a list of instance side methods

```
(Smalltalk at: #ZZFooBar) selectors
```

24.6 get a list of class side methods

```
(Smalltalk at: #ZZFooBar) class selectors
```

24.7 List methods defined in just this class - not inheritance chain

squeak - fails

```
(Smalltalk at: #ZZFooBar) methods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
```

```
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

24.8 List class side methods defined in just this class - not inheritance chain

squeak - fails

```
(Smalltalk at: #ZZFooBar) class methods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

24.9 List all instance methods - including inheritance chain

squeak - fails

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

24.10 List all class side methods - including inheritance chain

squeak - fails

```
(Smalltalk at: #ZZFooBar) class allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

24.11 Rename a method

FIXME

```
| oldName oldMethod newName |
oldName := #happy .
oldMethod := (Smalltalk at: #ZZFooBar) methodDict at: oldName.
newName := #hello.
(Smalltalk at: #ZZFooBar)
    compile: (oldMethod sourceCode copyReplaceAll: (oldName asString) with: newName asS
    classified: oldMethod protocolName.
(Smalltalk at: #ZZFooBar) removeSelector: oldName.
```

24.12 Rename a class side method

FIXME identical to rename an instance method , except add message class between them goodbye is both a symbol and a string in this example , depending on needs

```
| oldMethod newName |
oldMethod := (Smalltalk at: #ZZFooBar) class methodDict at: #goodbye.
newName := #farewell.
(Smalltalk at: #ZZFooBar) class
    compile: (oldMethod sourceCode copyReplaceAll: 'goodbye' with: newName asString)
    classified: oldMethod protocolName.
(Smalltalk at: #ZZFooBar) class removeSelector: #goodbye.
```

24.13 Delete a method

delete a method called goodbye

```
(Smalltalk at: #ZZFooBar) removeSelector: #goodbye.
```

24.14 Delete a class side method

FIXME

```
(Smalltalk at: #ZZFooBar) class removeSelector: #goodbye.
```

24.15 Delete a class

```
Smalltalk removeClassNamed: #ZZFooBar.
```

```
[ Smalltalk removeClassNamed: #ZZFooBar ]
    on: Error do: [ :ex | Transcript show: 'Error deleting class: ', ex messageText; cr
```

24.16 Delete a package

we had to ask organization , not to be confused with organisation which is different beast

```
Smalltalk organization removePackage: #ZZPackage.
```

24.17 Verify if class exists

```
Smalltalk includesKey: #ZZFooBar
```

25 FIXME FIXME FIXME we are upto here FIXME FIX ME

25.1 Create an instance side protocol

FIXME not sure if this is worth pursuing but hey..

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.2 Create an class side protocol

FIXME

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.3 Rename an instance side protocol

FIXME

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.4 Rename a class side protocol

FIXME

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.5 Delete an instance side protocol

FIXME

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.6 Delete a class side protocol

FIXME

```
(Smalltalk at: #ZZFooBar) allMethods do: [ :method |
    Transcript
        show: 'Selector: ', method selector asString;
```

```
        show: ' | Protocol: ', method protocolName asString;
        show: ' | Source: ', method sourceCode;
        cr
].
```

25.7 List all the protocols

FIXME what is a protocol ? squeak has no organization

```
(Smalltalk at: #ZZFooBar) organization protocols. "Instance-side protocols"
(Smalltalk at: #ZZFooBar) class organization protocols. "Class-side protocols"
```

25.8 change protocol under with a method is

squeak has no organization if protocol does not yet exist , it is created

```
(Smalltalk at: #ZZFooBar) organization
    classify: #hello
    under: 'new-protocol'
```

25.9 change classification of a method

FIXME Teach Smalltalk programming language as though everything done through the playground (also called workspace)

Be able to wield the full power of Smalltalk through the language completely without IDE or interface

Allows me to be able to save a text file and paste into playground and run !

==========

```
firstly open up any smalltalk image - first thing to do is save image as another name
this is because smalltalk insists on everything being mutable and saves randomly
so in order to keep original image clean we save as soon as startup
tried making certain files read only but corrupted ide programming interface

rule 1 : save a new image on start fresh image

configure pharo14.1 to start dirty image
```

```
configure pharo14 to start a clean development image

================================================================================

rule 2 :

================================================================================

"where-ever I say GT , I mean Glamorous Toolkit"

"topic : closures"

[:x | x + 1] value: 2 .

[:x :y | x + y] value: 2 value: 3.

"topic : classes"

"lets add a completely new class Pigeon"
Object subclass: #Pigeon.

"lets check it exists"
Pigeon browse.

"we find we do not see anything related to Pigeon this is because Pigeon class belongs

"we can coerce the symbol Pigeon to the corresponding class"
"FIXME this comparison did not work"
"#Pigeon asClass = Pigeon . "

"we can remove the pigeon class"
Smalltalk removeClassNamed: #Pigeon.

"how do we find if class Pigeon exists ? we check again Object class"
Smalltalk at: #Pigeon ifAbsent: [ ^ false ].
Smalltalk at: #Object ifAbsent: [ ^ false ].
Smalltalk at: #Pigeon ifPresent: [ ^ true ] ifAbsent: [ ^ false ].

"lets create Pigeon class again - to check no conflicts "
Object subclass: #Pigeon.
```

```
"lets check that Pigeon is identified as a class"
Pigeon class.

Pigeon browse.
"you may find you cannot see anything called Pigeon - it has no package and no category
"package is _UnpackagedPackage"

"lets give our pigeon class a package to live in "
birdsPackage := Smalltalk organization addPackage: #Birds.
birdsPackage addClass: Pigeon.
Pigeon browse.

" lets give our pigeon an instance variable - name"
Pigeon addInstVarNamed: #name.

"FIXME - this wont work at all ! lets add a method to Pigeon to say hello , the pigeon
(Smalltalk at: #Pigeon) compile: 'hello Transcript show: ''Pigeon says'' , name ; cr '

FIXME ... add a method to pigeon class ..
System Browser in pharo is called Calypso . all packages methods prefixed Cly presumabl

"lets make a pigeon and see if it squawks !"
p := Pigeon new.
p hello.
================================================================================

not sure how we interrupted execution of
================================================================================
ClySystemEnvironment we can get one from class instance method call
just a method call on the class itself , not an instance of a class

ClySystemEnvironment currentImage.

str := 'Object << #ZZFooBar
        layout: FixedLayout;
        traits: {};
        slots: { #cow . #milk };
        sharedVariables: {};
        sharedPools: {};
```

```
        tag: '''' ;
        package: ''ZZPackage'' '.
ClySystemEnvironment currentImage compileANewClassFrom: str notifying: nil startingFrom

A cheaper alternative to use Smalltalk classInstaller which didnt even know existed ! g

Smalltalk classInstaller
    make: [ :builder |
        builder
            superclass: Object;
            name: #ZZFooBar;
            slots: #(cow milk);
            classSlots: #();
            sharedPools: '';
            package: 'ZZPackage' ].


we can inspect the class
(Smalltalk at: #ZZFooBar) inspect. "Inspect the class"

ZZFooBar compile: 'hello3 ^ 3' classified: 'magic number3'.

(Smalltalk at: #ZZFooBar) instVarNames. "Returns #(#cow #milk)"
(Smalltalk at: #ZZFooBar) package name. "Returns 'ZZPackage'"

(Smalltalk at: #ZZFooBar) instVarNames. "Returns #(#cow #milk)"
(Smalltalk at: #ZZFooBar) package name. "Returns 'ZZPackage'"

"we added class side method test "
test
 ^ 'yes'

"running this should result in 'yes' "
ZZFooBar test.

"this just confirms that the system as whole is still working as it should"

"we can see Pigeon class now and a hello !"
================================================================================
```

```
ClassDescription >> #compile: sourceCode classified: protocol
we can now compile a method

ZZFooBar compile: 'hello3 ^ 3' classified: 'magic number3'.
================================================================================
Now for the class side we can see if we can get hold of ZZFooBar 's meta-class -
that should be the class side ?


str := 'Object << #ZZFooBar
        layout: FixedLayout;
        traits: {};
        slots: { #cow . #milk };
        sharedVariables: {};
        sharedPools: {};
        tag: '''' ;
        package: ''ZZPackage'' '.
ClySystemEnvironment currentImage compileANewClassFrom: str notifying: nil startingFro



================================================================================
how do we delete a method (or remove it )
or really how do we intercept what messages are causing things to actually happen ?

Smalltalk removeClassNamed: #ZZFooBar.
str := 'Object << #ZZFooBar
        layout: FixedLayout;
        traits: {};
        slots: { #cow . #milk };
        sharedVariables: {};
        sharedPools: {};
        tag: '''' ;
        package: ''ZZPackage'' '.
ClySystemEnvironment currentImage compileANewClassFrom: str notifying: nil startingFro
ZZFooBar compile: 'hello1 ^ 1' classified: 'magic number'.
ZZFooBar compile: 'hello2 ^ 2' classified: 'magic number'.
ZZFooBar compile: 'hello3 ^ 3' classified: 'odd number'.
```

```
how do we add a class side method ?
```

```
================================================================================
```

```
"we can list"
Smalltalk globals.
```

```
SmalltalkImage seems to be the entry point to the smalltalk image.
```

```
================================================================================
```

```
c := CircleMorph new openInHand.
b := BorderedMorph new openInHand .
```

```
================================================================================
```

```
"put pigeon into birds package "

"we can get a PackageOrganizer from Smalltalk"
"PackageOrganizer in charge of packages and package tags "
Smalltalk organization removePackage: #birds.
Smalltalk organization removePackage: #cows.

Smalltalk organization ensurePackage: 'birds'.
Smalltalk organization ensurePackage: 'fools' withTags: #( #foo) .
Smalltalk organization ensurePackage: 'fools' withTags: #( #foo #bar) .

"PackageTag has method addClass: "
"how do i make a package tag ? "

"xPackage addClass: c "

"lets add an initialize "

You can also directly execute a method, explicitly passing in the
```

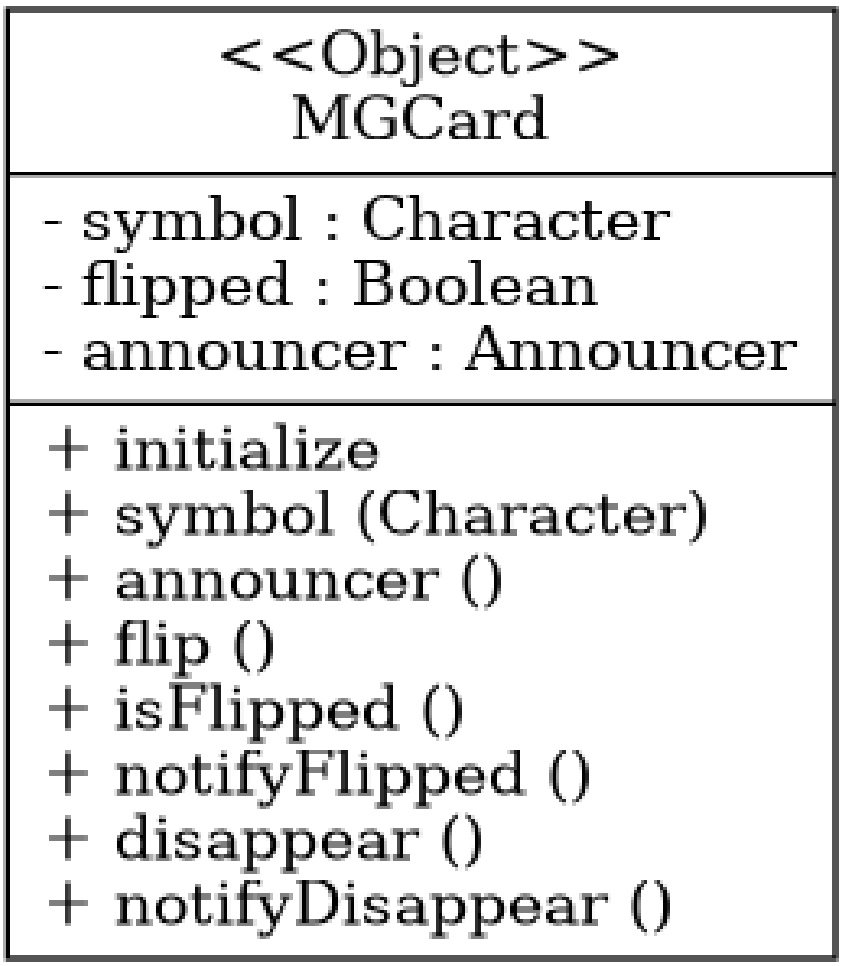
```
receiver and any arguments. Here we look up the hello method we
compiled earlier in the HelloWorld class. Then we directly execute the
method ( i.e., without any further lookup) with a Hello World instance
as the receover and an empty argument array:

method := #HelloWorld asClass>>#hello.
method valueWithReceiver: #HelloWorld asClass new arguments: #().

Smalltalk removeClassNamed: #Pigeon.


"we could also just slam a nil where HelloWorld would reside - this breaks things"
Smalltalk at: #HelloWorld put: nil.




"GT suggests
Object subclass: #HelloWorld  instanceVariableNames: ''  classVariableNames: ''  catego
"


"glamourous toolkit compiling and evaluating code "

Smalltalk compiler evaluate: '3 + 4'.

MGAlpha addClassVarNamed: 'ridiculous'.
MGAlpha addInstVarNamed: 'porkey'.

Cat
  compile: 'makeSound
    "Make Cat object make sound."
    Transcript show: ''Meow!''.'
  classified: 'actions'.

Class methods select: [:m | m selector beginsWith: 'subclass:'].


(Smalltalk at: #HelloWorld) compile: 'hello ^ ''hello'''.
```

```
#HelloWorld asClass compile: 'hello ^ ''hello'''.

((Smalltalk at: #HelloWorld) perform: #new) perform: #hello.

3 perform: #+ with: 4.

3 perform: #+ withArguments: {4}.



Metacello new
          baseline: 'BlocMemoryTutorial';
          repository: 'github://pharo-graphics/Bloc-Memory-Tutorial/src';
          load

MGGame withEmoji .

MGGameElement openWithNumber .
```

A graphical element will inherit from BlElement

```
BlElement << #MGAlpha
        slots: { #background };
        tag: 'Elements';
        package: 'Bloc-Memory'.

MGAlpha >> initialize [
    super initialize.
    self size: 80 @ 80.
    background := Color lightOrange.
    self background:  background.
    self geometry: BlCircleGeometry new.
    self addEventHandlerOn: BlClickEvent do: [ :anEvent | self click ]
]

MGAlpha >> click [
    background = Color lightOrange ifTrue:[ background := Color blue ] ifFalse:[ backgr
    self geometry: BlCircleGeometry new.
    "self addEventHandlerOn: BlClickEvent do: [ :anEvent | self click ]"
]
```

```
MGAlpha addClassVarNamed: 'ridiculous'.
MGAlpha addInstVarNamed: 'porkey'.


  (add-to-list 'org-structure-template-alist
               '("s" "#+NAME: ?\n#+BEGIN_SRC \n\n#+END_SRC"))
;; in org mode
;; press <s  TAB should give
"#+NAME:"
"#+BEGIN_SRC"
"#+END_SRC"


(ql:quickload :dml)
(in-package :dml)

;; MG memory game
(dml-create-graph "mgcard-class" ()

  ;; mgcard class
  (with-method ("+ initialize"
                "+ symbol (Character)"
                "+ announcer ()"
                "+ flip ()"
              "+ isFlipped ()"
              "+ notifyFlipped ()"
                "+ disappear ()"
                "+ notifyDisappear ()")
    (full-class "MGCard"
                "Object"
                (attributes "- symbol : Character"
                            "- flipped : Boolean"
                            "- announcer : Announcer"
                            ))))
```

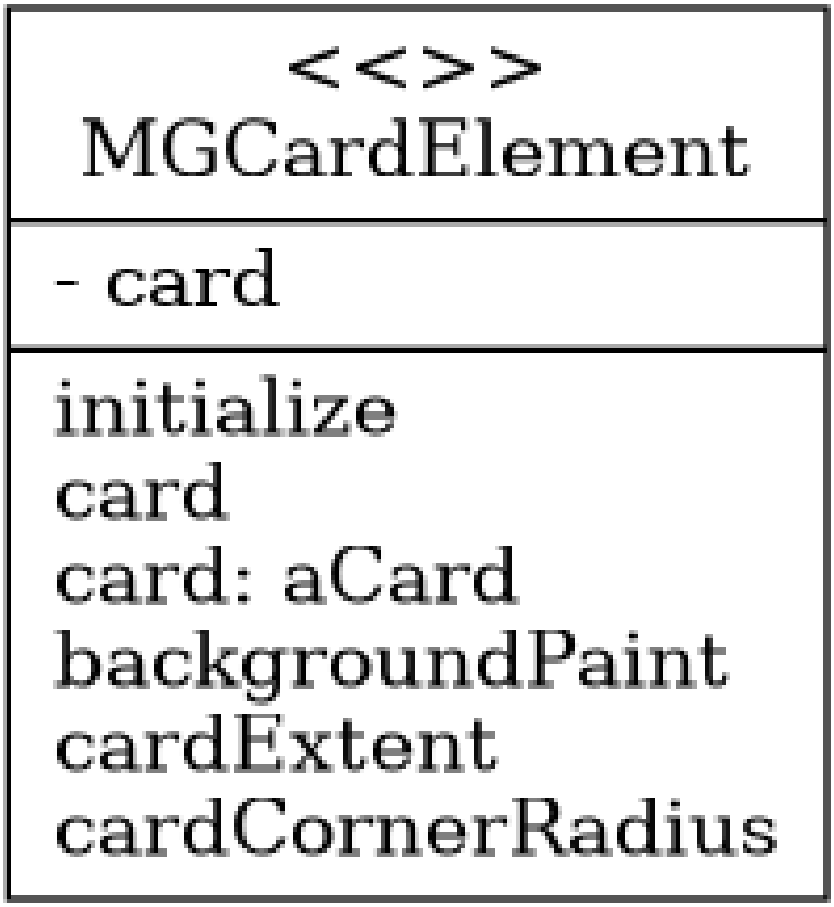



```
Class {
        #name : 'MGCard',
        #superclass : 'Object',
        #instVars : [
                'symbol',
                'flipped',
                'announcer'
        ],
        #category : 'Bloc-Memory-Model',
        #package : 'Bloc-Memory',
        #tag : 'Model'
}
```

```
MGCard >> announcer [
 ^ announcer ifNil: [ announcer := Announcer new ]
]

MGCard >> disappear [
self notifyDisappear
]

MGCard >> flip [
 flipped := flipped not.
 self notifyFlipped.
]

MGCard >> initialize [
 super initialize.
 flipped := false.
]

MGCard >> isFlipped [
 ^ flipped
]

MGCard >> notifyDisappear [
 self announcer announce: MGCardDisappearAnnouncement new
]

MGCard >> notifyFlipped [
 self announcer announce: MGCardFlippedAnnouncement new
]

MGCard >> printOn: aStream [
aStream
nextPutAll: 'Card';
nextPut: Character space;
nextPut: $( ;
nextPut: self symbol;
nextPut: $)
]
```

```
MGCard >> symbol [
 ^ symbol
]

MGCard >> symbol: aCharacter [
 symbol := aCharacter.
]
```

```
(ql:quickload :dml)
(in-package :dml)

;; MG memory game
(dml-create-graph "mgcard-element-class" ()

  ;; mgcard class
  (with-method ("initialize"
                "card"
                "card: aCard"
                "backgroundPaint"
                "cardExtent"
                "cardCornerRadius")
    (full-class "MGCardElement"
                ""
                (attributes "- card "
                            ))))
```



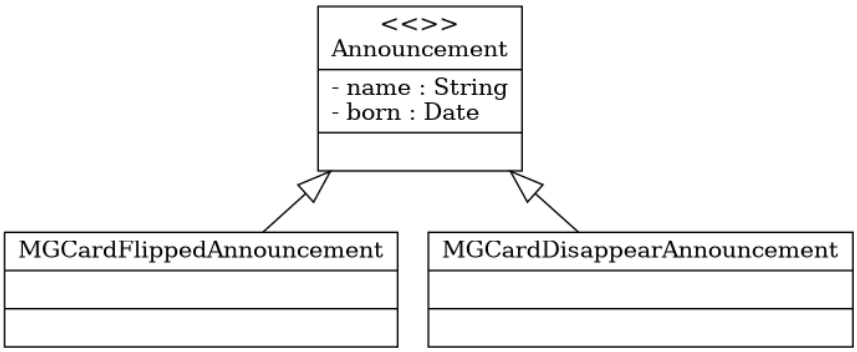

```
"
In Bloc, BlElements draw themselves onto the integrated canvas of the inspector as we inspect them, take a look at our element by executing this (See Figure 3-1).
‘‘‘
MGCardElement new inspect
‘‘‘
"
Class {
        #name : ’MGCardElement’,
        #superclass : ’BlElement’,
        #instVars : [
’card’
```

```
],
        #category : 'Bloc-Memory-Elements',
        #package : 'Bloc-Memory',
        #tag : 'Elements'
    }
MGCardElement >> card [
^ card
]

MGCardElement >> card: aMgCard [
    card  := aMgCard
]


MGCardElement >> backgroundPaint [
    "Return a BlPaint that should be used as a background (fill)
of both back and face sides of the card. Colors are polymorphic
with BlPaint and therefore can be used too."
    ^ Color pink darker
]


MGCardElement >> initialize [
    super initialize.
"    self size: 80 @ 80. " "replaced with cardExtent"
    self size: self cardExtent.
    "A BlBackground is needed for the #background: method, but the
BlPaint
is polymorphic with BlBackground and therefore can be used too."
    self background: self backgroundPaint.

    " no geometry to circle to rounded rectangle"
    " self geometry: BlCircleGeometry new. "
    self geometry: (BlRoundedRectangleGeometry cornerRadius: self cardCornerRadius ).
    self card: (MGCard new symbol: $a)
]


MGCardElement >> cardExtent [
^ 80@80
```

```
]

MGCardElement >> cardCornerRadius [
^ 12
]

"cardbackForm bitmap from bloc-memory game"
"just get the code"
```

Announcements

```
(ql:quickload :dml)
(in-package :dml)

;; MG memory game
(dml-create-graph "mgcard-announcement-classes" ()

  (-genby-*
     (full-class "Announcement"
                 ""
                 (attributes "- name : String"
                             "- born : Date"))
     (full-class "MGCardFlippedAnnouncement")
     (full-class "MGCardDisappearAnnouncement")))
```




```
Class {
        #name : 'MGCardDisappearAnnouncement',
        #superclass : 'Announcement',
```

```
        #category : 'Bloc-Memory-Events',
        #package : 'Bloc-Memory',
        #tag : 'Events'
}

Class {
        #name : 'MGCardFlippedAnnouncement',
        #superclass : 'Announcement',
        #category : 'Bloc-Memory-Events',
        #package : 'Bloc-Memory',
        #tag : 'Events'
}
```

Package.st file contains name of package

```
Package { #name : 'Bloc-Memory' }
```

Hidden .properties file - tonel

```
{
        #format : #tonel
}
```

pharo bloc memory game tutorial
bloc is low level graphics
brick is widget library built on top
tangle C-c C-v C-t C-c C-v C-a org-babel-sha1-hash C-c C-v C-b org-babel-execute-buffer C-c C-v C-c org-babel-check-src-block C-c C-v C-d org-babel-demarcate-block C-c C-v C-e org-babel-execute-maybe C-c C-v C-f org-babel-tangle-file C-c C-v TAB org-babel-view-src-block-info C-c C-v C-j org-babel-insert-header-arg C-c C-v C-l org-babel-load-in-session C-c C-v C-n org-babel-next-src-block C-c C-v C-o org-babel-open-src-block-result C-c C-v C-p org-babel-previous-src-block C-c C-v C-r org-babel-goto-named-result C-c C-v C-s org-babel-execute-subtree C-c C-v C-t org-babel-tangle C-c C-v C-u org-babel-goto-src-block-head C-c C-v C-v org-babel-expand-src-block C-c C-v C-x org-babel-do-key-sequence-in-edit-buffer C-c C-v C-z org-babel-switch-to-session C-c C-v I org-babel-view-src-block-info C-c C-v a org-babel-sha1-hash C-c C-v b org-babel-execute-buffer C-c C-v c org-babel-check-src-block C-c C-v d org-babel-demarcate-block C-c C-v e org-babel-execute-maybe C-c C-v f org-babel-tangle-file C-c C-v g org-babel-goto-named-src-block C-c C-v h org-babel-describe-bindings C-c C-v i org-

babel-lob-ingest C-c C-v j org-babel-insert-header-arg C-c C-v k org-babel-remove-result-one-or-many C-c C-v l org-babel-load-in-session C-c C-v n org-babel-next-src-block C-c C-v o org-babel-open-src-block-result C-c C-v p org-babel-previous-src-block C-c C-v r org-babel-goto-named-result C-c C-v s org-babel-execute-subtree C-c C-v t org-babel-tangle C-c C-v u org-babel-goto-src-block-head C-c C-v v org-babel-expand-src-block C-c C-v x org-babel-do-key-sequence-in-edit-buffer C-c C-v z org-babel-switch-to-session-with-code

C-c " a orgtbl-ascii-plot C-c " g org-plot/gnuplot

C-c C-M-l org-insert-all-links C-c C-M-w org-refile-reverse C-c M-b org-previous-block C-c M-f org-next-block C-c M-l org-insert-last-stored-link C-c M-w org-refile-copy

C-c C-x C-a org-archive-subtree-default C-c C-x C-b org-toggle-checkbox C-c C-x C-c org-columns C-c C-x C-d org-clock-display C-c C-x C-f org-emphasize C-c C-x TAB org-clock-in C-c C-x C-j org-clock-goto C-c C-x C-l org-latex-preview C-c C-x C-n org-next-link C-c C-x C-o org-clock-out C-c C-x C-p org-previous-link C-c C-x C-q org-clock-cancel C-c C-x C-r org-toggle-radio-button C-c C-x C-s org-archive-subtree C-c C-x C-t org-toggle-time-stamp-overlays C-c C-x C-u org-dblock-update C-c C-x C-v org-toggle-inline-images C-c C-x C-w org-cut-special C-c C-x C-x org-clock-in-last C-c C-x C-y org-paste-special C-c C-x C-z org-resolve-clocks C-c C-x ! org-reload C-c C-x , org-timer-pause-or-continue C-c C-x - org-timer-item C-c C-x . org-timer C-c C-x 0 org-timer-start C-c C-x ; org-timer-set-timer C-c C-x < org-agenda-set-restriction-lock C-c C-x > org-agenda-remove-restriction-lock

C-c C-x @ org-cite-insert C-c C-x A org-archive-to-archive-sibling C-c C-x E org-inc-effort C-c C-x G org-feed-goto-inbox C-c C-x I org-info-find-node C-c C-x P org-set-property-and-value C-c C-x [org-reftex-citation C-c C-x \ org-toggle-pretty-entities C-c C-x _ org-timer-stop C-c C-x a org-toggle-archive-tag C-c C-x b org-tree-to-indirect-buffer C-c C-x c org-clone-subtree-with-time-shift C-c C-x d org-insert-drawer C-c C-x e org-set-effort C-c C-x f org-footnote-action C-c C-x g org-feed-update-all C-c C-x o org-toggle-ordered-property C-c C-x p org-set-property C-c C-x q org-toggle-tags-groups C-c C-x v org-copy-visible C-c C-x x org-dynamic-block-insert-dblock

C-c C-v C-M-h org-babel-mark-block

C-c C-x C-M-v org-redisplay-inline-images C-c C-x M-w org-copy-special

#+BEGIN$_{\text{SRC}}$ c

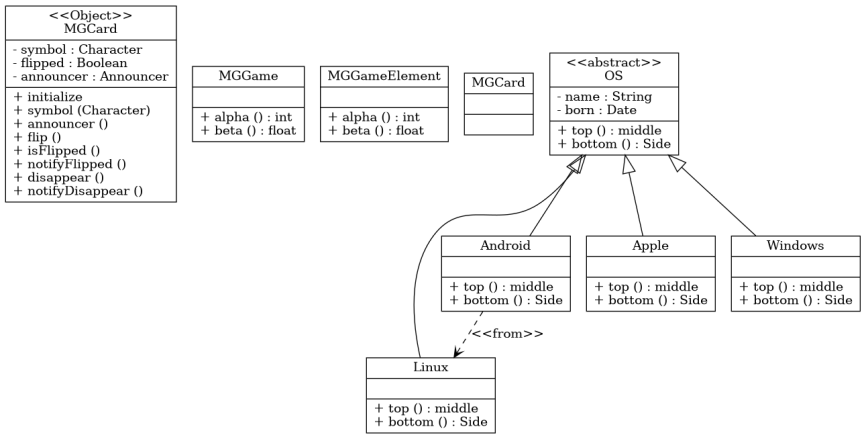
##+header: :var message="Hello World!"

```
(ql:quickload :dml)   ;
```

```
(in-package :dml)

(format t "hello world")


NIL

Metacello new
baseline: 'BlocMemoryTutorial';
repository: 'github://pharo-graphics/Bloc-Memory-Tutorial/src';
load

(ql:quickload :dml)   ;
(in-package :dml)

(dml-create-graph "os-class" ()
  (with-method ("+ play () : Love" "+ work () : Hate")
    (-genby-*
     (full-class "OS"
                 "abstract"
                 (attributes "- name : String"
                             "- born : Date"))
     (full-class "Linux")
     (full-class "Android")
     (full-class "Apple")
     (full-class "Windows"))
    (-dep- "from"
           (@name "Android")
           (@name"Linux"))))


(ql:quickload :dml)
(in-package :dml)

;; MG memory game
(dml-create-graph "mgdemo-classes" ()

  ;; mgcard class
  (with-method ("+ initialize"
  "+ symbol (Character)"
  "+ announcer ()"
```

```
"+ flip ()"
     "+ isFlipped ()"
     "+ notifyFlipped ()"
"+ disappear ()"
"+ notifyDisappear ()")
  (full-class "MGCard"
"Object"
(attributes "- symbol : Character"
     "- flipped : Boolean"
     "- announcer : Announcer"
     )))


;; mggame class
(with-method ("+ alpha () : int" "+ beta () : float")
  (full-class "MGGame"))

;; mggameelement class
(with-method ("+ alpha () : int" "+ beta () : float")
  (full-class "MGGameElement"))




(full-class "MGCard")
(with-method ("+ top () : middle" "+ bottom () : Side")
  (-genby-*
   (full-class "OS"
 "abstract"
 (attributes "- name : String"
      "- born : Date"))
   (full-class "Linux")
   (full-class "Android")
   (full-class "Apple")
   (full-class "Windows"))
  (-dep- "from"
  (@name "Android")
  (@name"Linux"))))
```

`NIL`



25.10 Lets make a circle

25.11 Lets make a circle