Anchors

^	Start of line +
\A	Start of string +
$	End of line +
\Z	End of string +
\b	Word boundary +
\B	Not word boundary +
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\xhh	Hexadecimal character hh
\Oxxx	Octal character xxx

POSIX Character Classes

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Assertions

?=	Lookahead assertion +
?!	Negative lookahead +
?<=	Lookbehind assertion +
?!= or ?<!	Negative lookbehind +
?>	Once-only Subexpression
?()	Condition [if then]
?()|	Condition [if then else]
?#	Comment

Note Items marked + should work in most

Sample Patterns

([A-Za-z0-9-]+)	Letters, numbers and hyphens
(\d{1,2}\/\d{1,2}\/\d{4})	Date (e.g. 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	jpg, gif or png image
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Any number from 1 to 50 inclusive
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Valid hexadecimal colour code
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	8 to 15 character string with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
(\<(/?[^\>]+)\>)	HTML Tags

Note *These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.*

Quantifiers

*	0 or more +
*?	0 or more, ungreedy +
+	1 or more +
+?	1 or more, ungreedy +
?	0 or 1 +
??	0 or 1, ungreedy +
{3}	Exactly 3 +
{3,}	3 or more +
{3,5}	3, 4 or 5 +
{3,5}?	3, 4 or 5, ungreedy +

Special Characters

\	Escape Character +
\n	New line +
\r	Carriage return +
\t	Tab +
\v	Vertical tab +
\f	Form feed +
\a	Alarm
[\b]	Backspace
\e	Escape
\N{name}	Named Character

String Replacement (Backreferences)

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string
$_	Entire input string
$$	Literal "$"

Ranges

.	Any character except new line (\n) +
(a|b)	a or b +
(...)	Group +
(?:...)	Passive Group +
[abc]	Range (a or b or c) +
[^abc]	Not a or b or c +
[a-q]	Letter between a and q +
[A-Q]	Upper case letter + between A and Q +
[0-7]	Digit between 0 and 7 +
\n	nth group/subpattern +

Note *Ranges are inclusive.*

Pattern Modifiers

g	Global match
i	Case-insensitive
m	Multiple lines
s	Treat string as single line
x	Allow comments and white space in pattern
e	Evaluate replacement
U	Ungreedy pattern

Metacharacters (must be escaped)

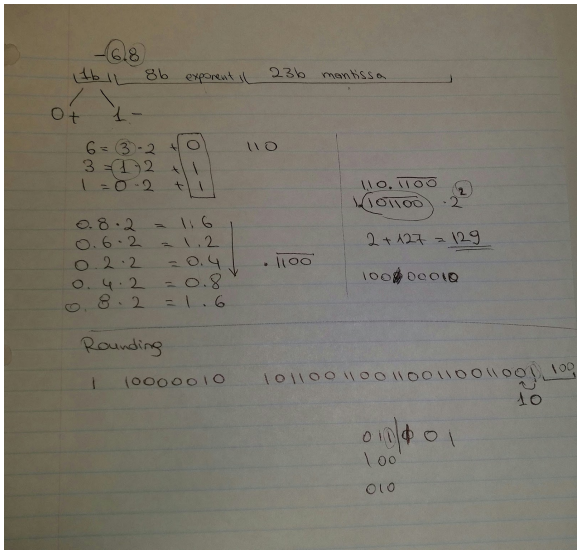
^	[	.
$	{	*
(	\	+
)	|	?
<	>	

IEEE 754 Floating Points

1b 8b 23b - [Pos/Neg] [Exponent - 127] [Fraction]

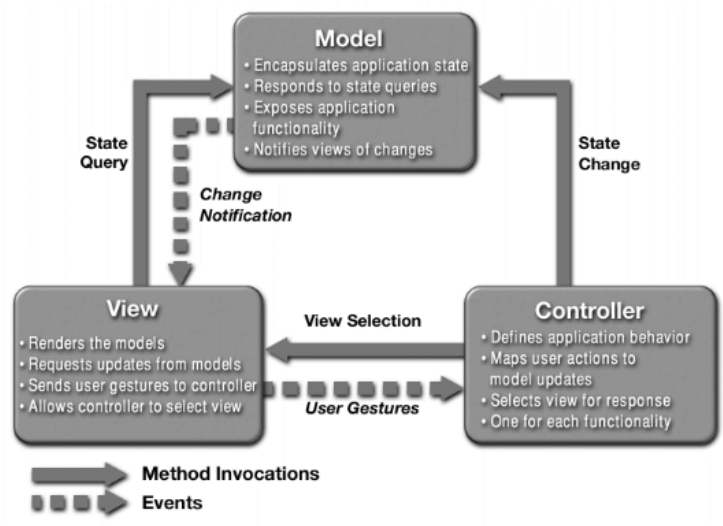
Rounding:

- 24th bit is 0 do nothing
- Next bit is 1 followed by 10, 01, 11 round up
- Next bit is 1 followed by 100
    - If 23rd bit is 1 round up
    - If 24th bit is 0 do nothing



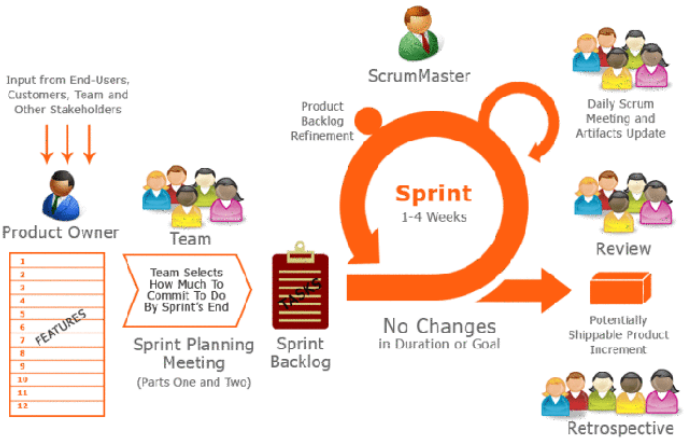
UML Structure: If not specified, Java will default to Package Private

Name
instanceVariable: type
+ Constructor (var1: type, var2: type) + publicMethod(): returnType - privateMethod(): returnType # protectedMethod(): returnType ~ packageMethod(): returnType + staticMethod(): returnType



MVC Notes:

- Controller gets the View and Model
- View implements Observer
- Controller implements Actionlistener
- Model extends Observable
- View Actionlistener is controller
- Model observer is view



Scrum Notes:

- Product Owner: Knows what the product is supposed to do. They have the product vision, and are responsible for communicating this clearly to the Team
- Product Backlog This is a prioritized list of User Stories that they would like to see in the system
- User Story is essentially an atomic feature from the perspective of a user. It usually captures a single interaction with the system written in the language of the users.
- Scrum Master: Serves both the Product Owner and the Team, knows the Scrum process, ensures that the process is taking place. Basically, they make sure everyone is following the above diagram.

- **Sprint Planning:** A meeting with PO, SM and Team. Take highest priority user stories and create the **Sprint Backlog**.

```
Sprint Backlog (Highest Priority Prefix of Product Backlog)

Priority          User Story     Estimate (added by team)
--------          ----------     --------
   1              story 7           1 day
   2              story 9           2 days
   3              story 3           1 day
   4              story 12          4 days

!! Can't take any more work for this Sprint.
```

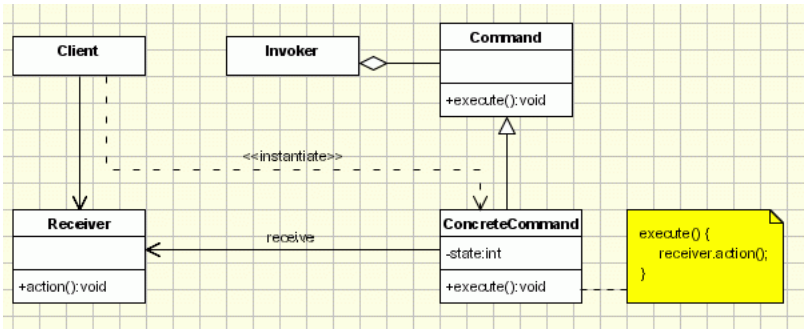
- **Sprint (1-3 weeks):** Team works on delivering the Sprint Backlog. They architect, code, test, document etc. They also hold Daily Scrum Meeting.
- **Daily Scrum Meeting:** A meeting with SM and Team. 15 Minute standup meeting, each team member says

```
What I did yesterday.
What I am doing today.
Obstacles I face.
```

- **Sprint Review:** A meeting with PO, SM and Team. Show off product to PO. Acceptance test as well as discussion of Scrum Process improvement.
- **Sprint Retrospective** (more later)

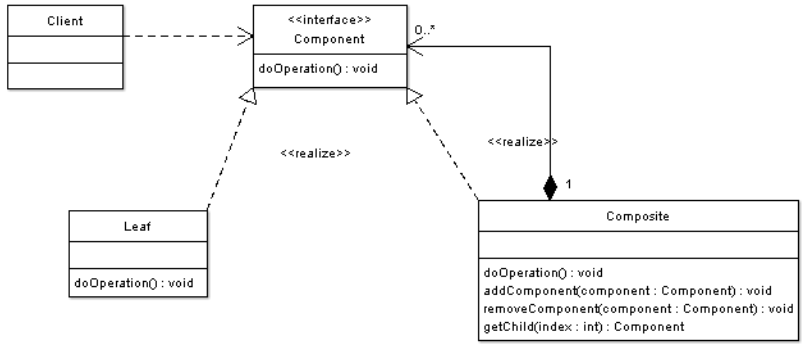
File IO:

- FileReader, System.in.read, Scanner(System.in) are inputs which can only read byte by byte (1 char)
- Create this with a BufferedReader to read line by line
- FileWriter can write byte by byte (1 char)
- PrintWriter can print full lines



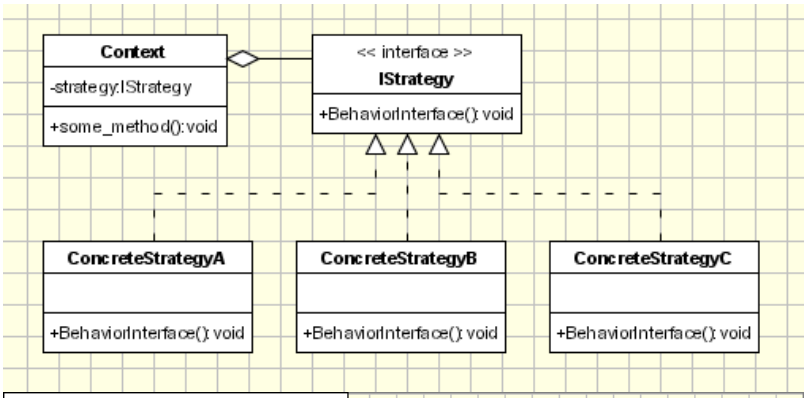
Command:

- Abstract command as a parent
- Each command has an execution
- Can store a list of command and execute them later



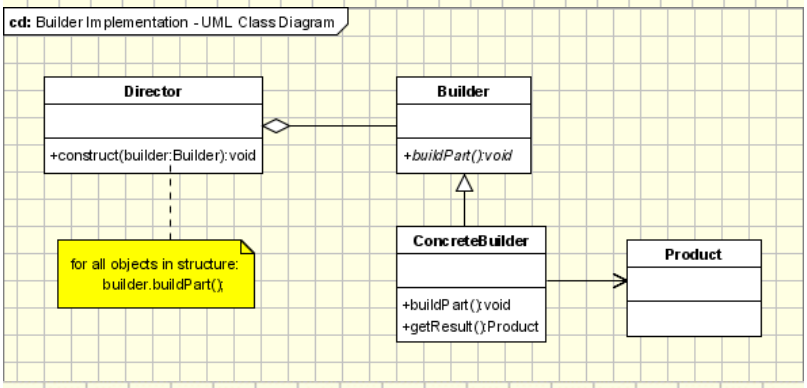
Composite:

- Essentially create a linked list of object
- Keep a “next” object as reference, keep all objects in a list
- Able to get the child object in the list
- Actually does the work



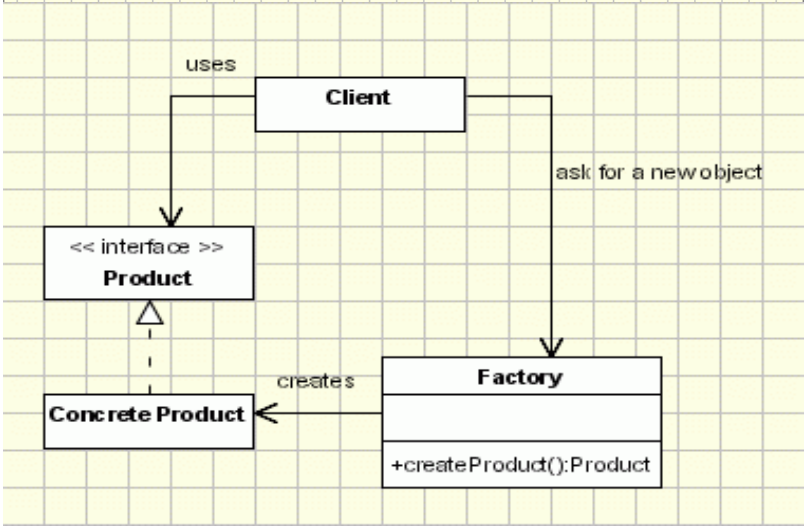
Strategy:

- Interface a strategy to an object
- Gives a method for solving the same problem
- Gives instructions to do the work
- Eg. Sort Algorithm choosing



Builder:

- Given instructions to create an object
- Can call build to get the objects with given specs
- Other builders can inherit the base builder to give a specific output



Factory:

- Given a product name, can create the related object
- No additional attributes or details are passed

Compiling RegEx:

```
Pattern pattern = Pattern.compile( String RegLine );
Matcher m = pattern.matcher( String LineToCheck );
m.matches( );
```

```
p = Pattern.compile("CSC([0-9][0-9][0-9])H1(F|S)");
m = p.matcher("CSC207H1S");
System.out.println(m.matches());
System.out.println(m.group(0));// the entire string
System.out.println(m.group(1));// the first group: 207
System.out.println(m.group(2));// the second group: S
```

FSM with files:

```
While (not EOF)
        Read Line
                Switch (state)
                        State 1:
                                Do stuff
                                State = 2
                                break
                        State 2
                                So stuff
                                State = 3
                                Break
                ETC.
```

```
0[00000000]00000000000000000000000 = 0

0[11111111]00000000000000000000000 = Infinity

0[11111111]mmmmmmmmmmmmmmmmmmmmmmm = NaN (Not a Number)
 where mmmmmmmmmmmmmmmmmmmmmmm != 00000000000000000000000

0[00000000]mmmmmmmmmmmmmmmmmmmmmmm = Denormalized Numbers
 where mmmmmmmmmmmmmmmmmmmmmmm != 00000000000000000000000

0[eeeeeeee]mmmmmmmmmmmmmmmmmmmmmmm = Normalized Numbers
 where eeeeeeee!=00000000 and eeeeeeee!=11111111
```

we make you struggle, know the right struggle to have. Then you know the material.

-Arnold 2016