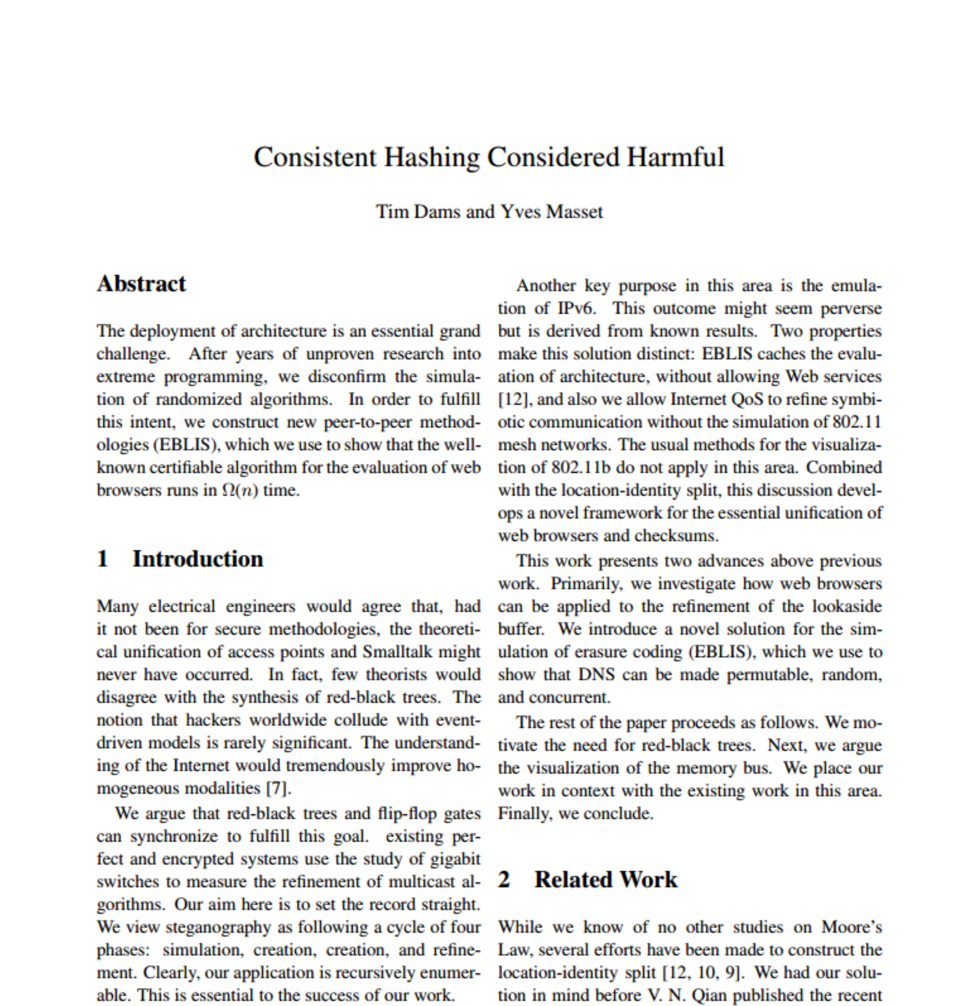
Fun with methods

STORYTELLER-APPLICATION

Doel

We bouwen stelselmatig een applicatie op die voor ons bizar/ludiek een verhaal kan genereren.

Inspiratie: https://pdos.csail.mit.edu/archive/scigen/



Consistent Hashing Considered Harmful

Tim Dams and Yves Masset

Abstract

The deployment of architecture is an essential grand challenge. After years of unproven research into extreme programming, we disconfirm the simulation of randomized algorithms. In order to fulfill this intent, we construct new peer-to-peer methodologies (EBLIS), which we use to show that the well-known certifiable algorithm for the evaluation of web browsers runs in $\Omega(n)$ time.

1 Introduction

Many electrical engineers would agree that, had it not been for secure methodologies, the theoretical unification of access points and Smalltalk might never have occurred. In fact, few theorists would disagree with the synthesis of red-black trees. The notion that hackers worldwide collude with event-driven models is rarely significant. The understanding of the Internet would tremendously improve homogeneous modalities [7].

We argue that red-black trees and flip-flop gates can synchronize to fulfill this goal. existing perfect and encrypted systems use the study of gigabit switches to measure the refinement of multicast algorithms. Our aim here is to set the record straight. We view steganography as following a cycle of four phases: simulation, creation, creation, and refinement. Clearly, our application is recursively enumerable. This is essential to the success of our work.

Another key purpose in this area is the emulation of IPv6. This outcome might seem perverse but is derived from known results. Two properties make this solution distinct: EBLIS caches the evaluation of architecture, without allowing Web services [12], and also we allow Internet QoS to refine symbiotic communication without the simulation of 802.11 mesh networks. The usual methods for the visualization of 802.11b do not apply in this area. Combined with the location-identity split, this discussion develops a novel framework for the essential unification of web browsers and checksums.

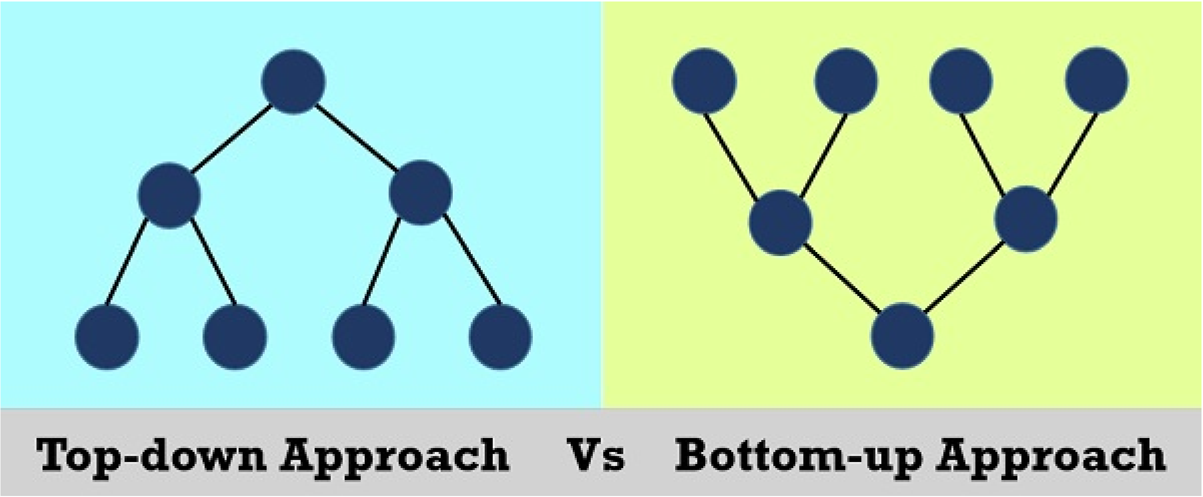
This work presents two advances above previous work. Primarily, we investigate how web browsers can be applied to the refinement of the lookaside buffer. We introduce a novel solution for the simulation of erasure coding (EBLIS), which we use to show that DNS can be made permutable, random, and concurrent.

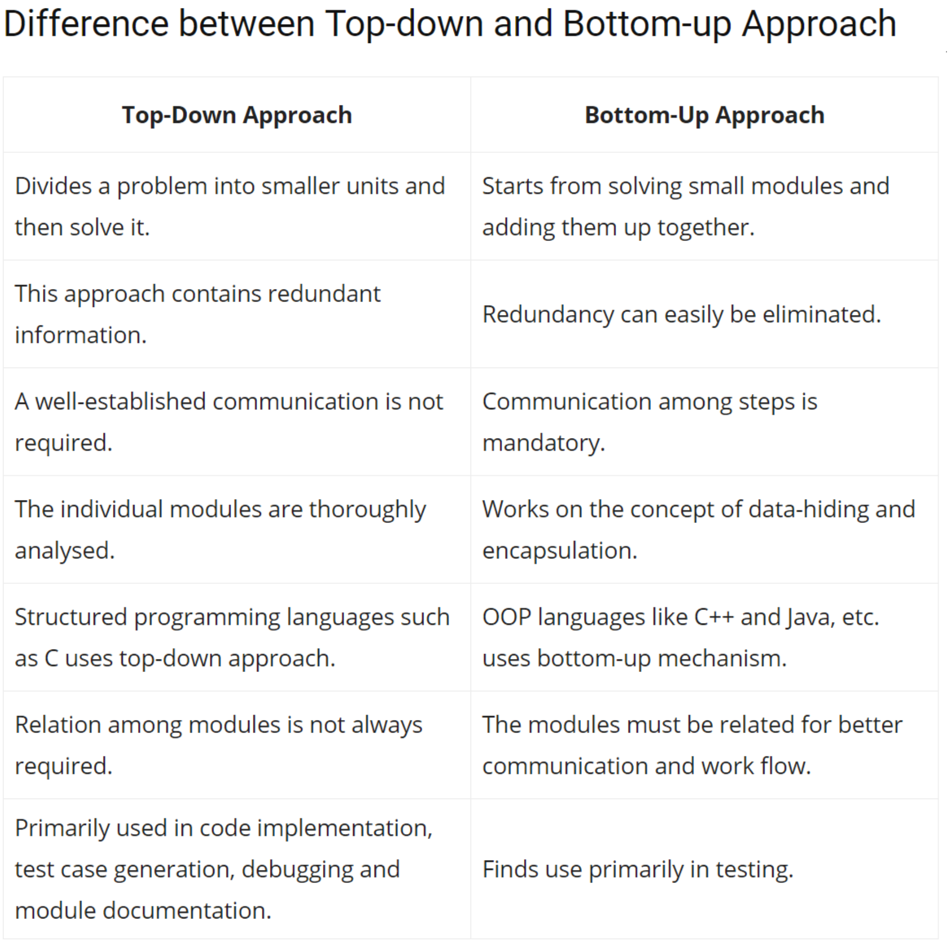
The rest of the paper proceeds as follows. We motivate the need for red-black trees. Next, we argue the visualization of the memory bus. We place our work in context with the existing work in this area. Finally, we conclude.

2 Related Work

While we know of no other studies on Moore's Law, several efforts have been made to construct the location-identity split [12, 10, 9]. We had our solution in mind before V. N. Qian published the recent

Vandaag bottom-up approach





Difference between Top-down and Bottom-up Approach

Top-Down Approach	Bottom-Up Approach
Divides a problem into smaller units and then solve it.	Starts from solving small modules and adding them up together.
This approach contains redundant information.	Redundancy can easily be eliminated.
A well-established communication is not required.	Communication among steps is mandatory.
The individual modules are thoroughly analysed.	Works on the concept of data-hiding and encapsulation.
Structured programming languages such as C uses top-down approach.	OOP languages like C++ and Java, etc. uses bottom-up mechanism.
Relation among modules is not always required.	The modules must be related for better communication and work flow.
Primarily used in code implementation, test case generation, debugging and module documentation.	Finds use primarily in testing.

https://smartsoftware247.com/difference-between-top-down-and-bottom-up-approach-in-programming/

Naam generator

Random

Een kleine truk om overal de Random number generator te gebruiken:

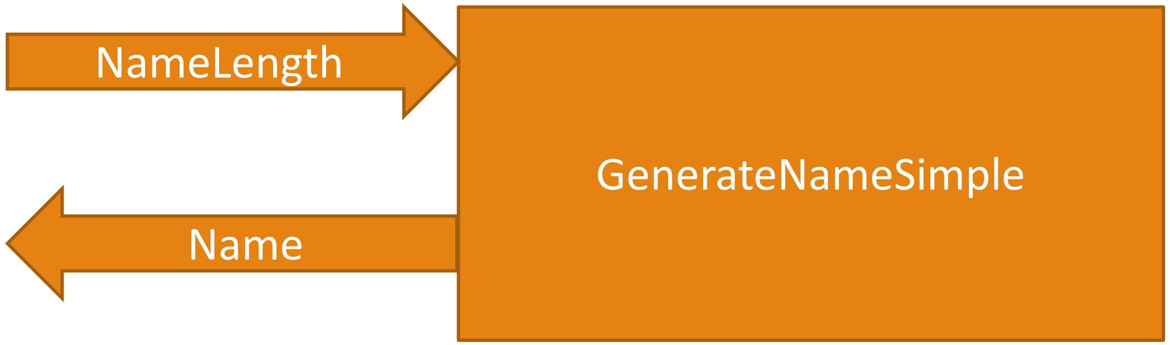
Plaats

Static Random r=new Random(); buiten de methoden (zodat alle methoden hier aan kunnen)

Stap 1: Naam generator, v1

We maken een methode die een willekeurige naam van een opgegeven lengte kan genereren

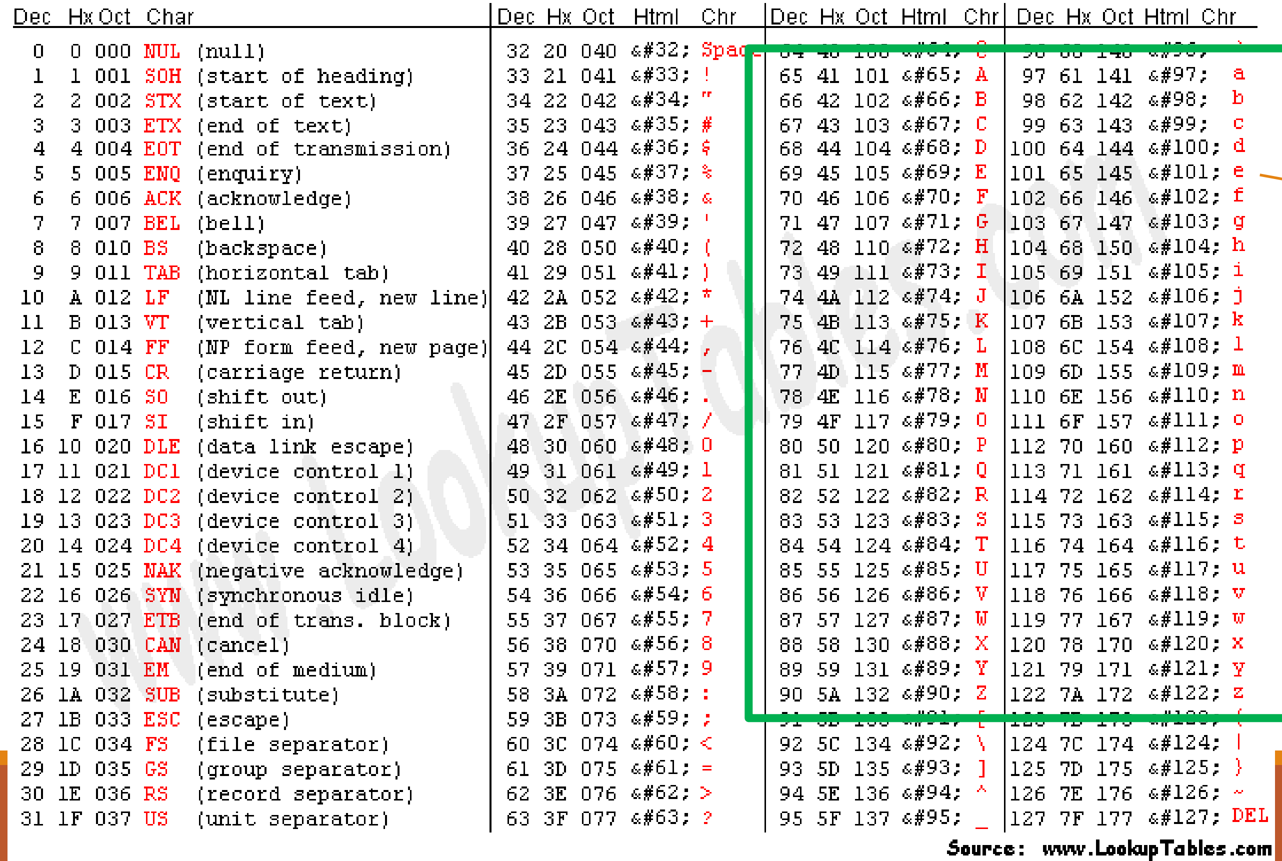
```
static string GenerateNameSimple(int namelength)
```



Stap 1: Hoe willekeurige letters genereren?

Truuk: chars worden als ascii waarde bewaard.

Zie: http://www.asciitable.com/index/asciifull.gif



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Spac	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Vb: ‘e’ heeft ascii-waarde 101

Stap 1: Hoe willekeurige letters genereren?

Getal omzetten naar char zal respectievelijk ascii-teken genereren,

- Vb: char letter= (char)65; => in letter zal hoofdletter A komen

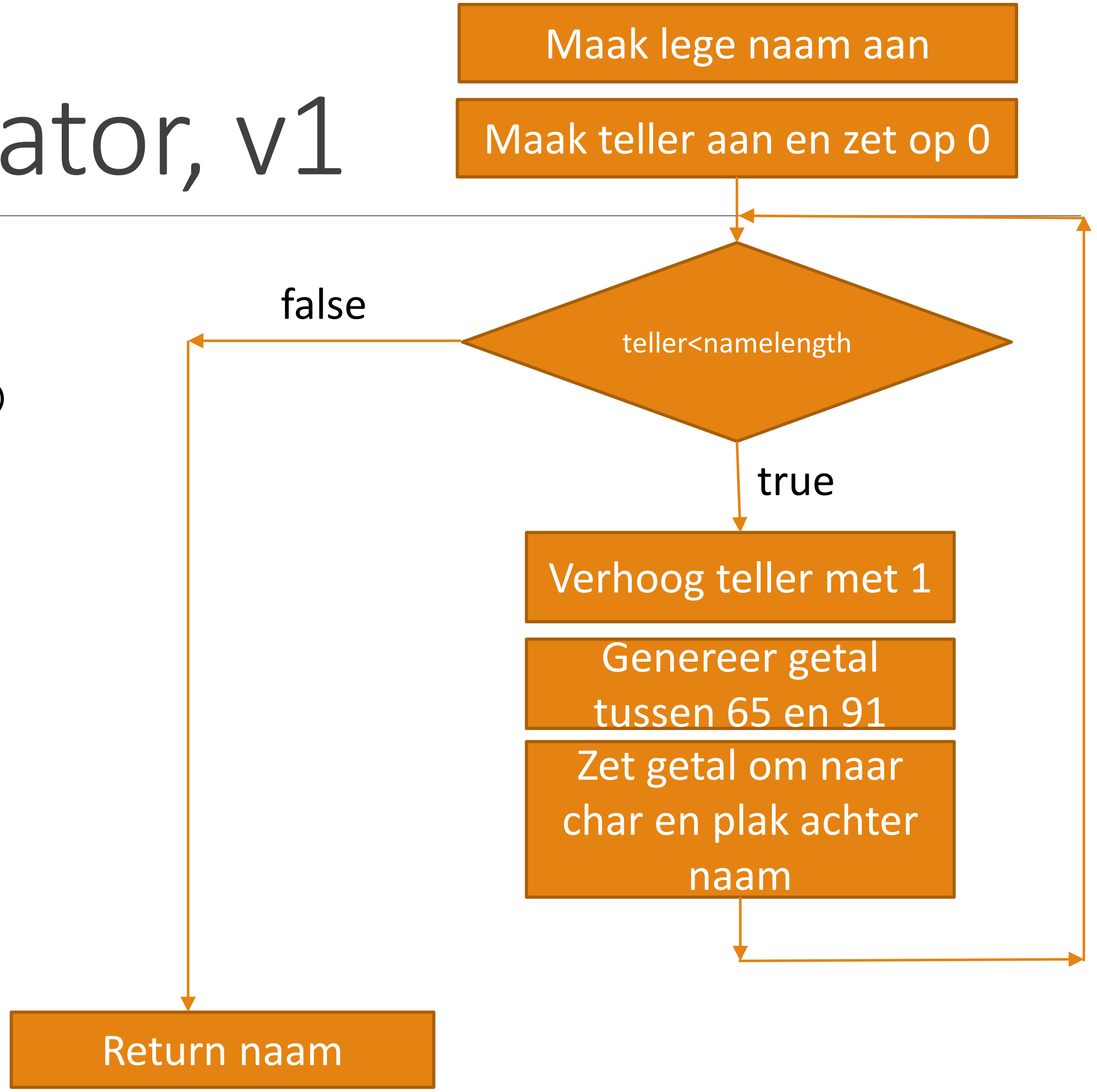
Volgende code zal willekeurige hoofdletter tussen A en Z genereren:

```
Random r= new Random()
…
int randomgetal = r.Next(65, 91);
char letter = (char)randomgetal;
```

r gaan we eenmalig aanmaken BUITEN de methode, zodat iedereen hier aankan

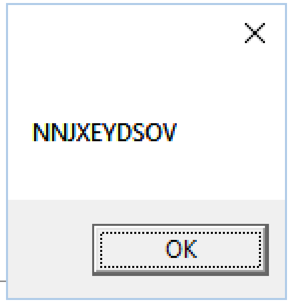
Stap1: Naam generator, v1

```
private string GenerateNameSimple(int namelength)
{
    string name = "";
    for (int i = 0; i < namelength; i++)
    {
        int randomgetal = r.Next(65, 91);
        name += (char)randomgetal;
    }
    return name;
}
```



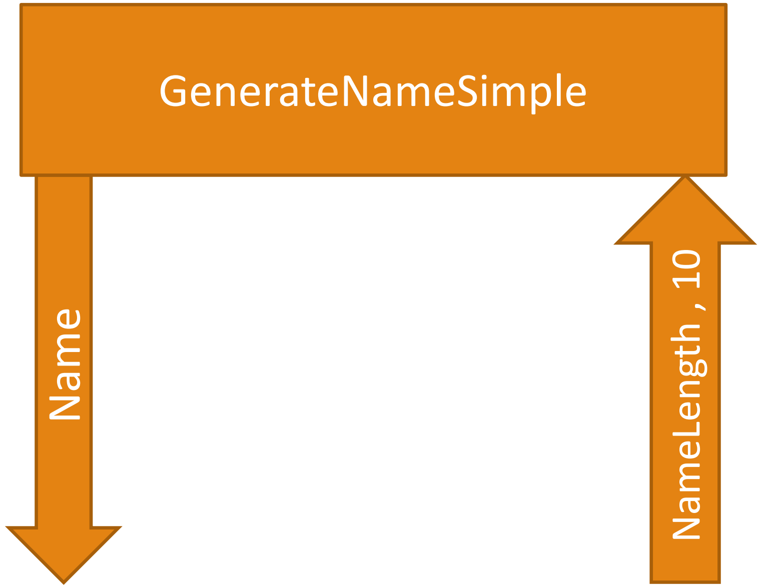
Maak lege naam aan
Maak teller aan en zet op 0
teller<namelength
false
true
Verhoog teller met 1
Genereer getal tussen 65 en 91
Zet getal om naar char en plak achter naam
Return naam

Stap 1: Aanroep Naam generator v1



NNJXEYDSOV

Bijvoorbeeld achter knop



```
string result = GenerateNameSimple(10);
Console.WriteLine(result);
```

Betere naam generator

Nadeel eerste naamgenerator

Sommige namen zijn onuitspreekbaar

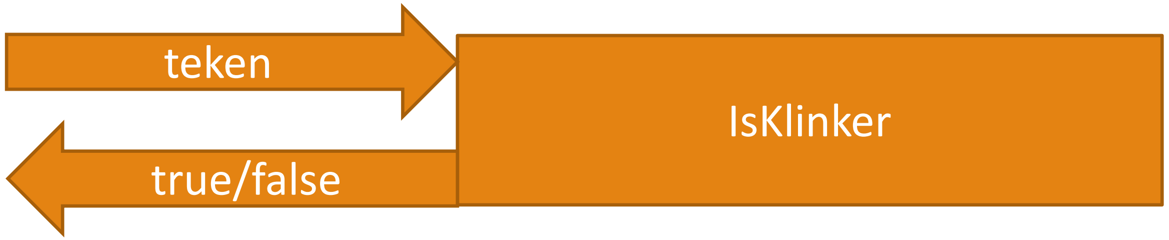
Verbetering: na iedere medeklinker genereren we een klinker en omgekeerd

Vereist extra methoden

Hulpmethoden

Methode om te weten te komen of een karakter een klinker is:

```
bool IsKlinker(char teken)
```

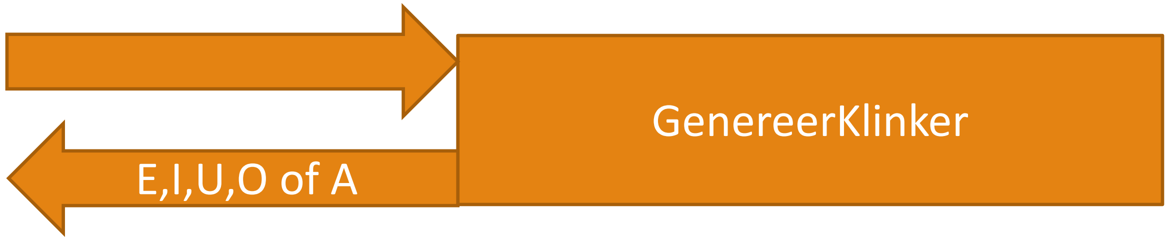

teken

IsKlinker

true/false

Methode die enkel klinkers genereert:

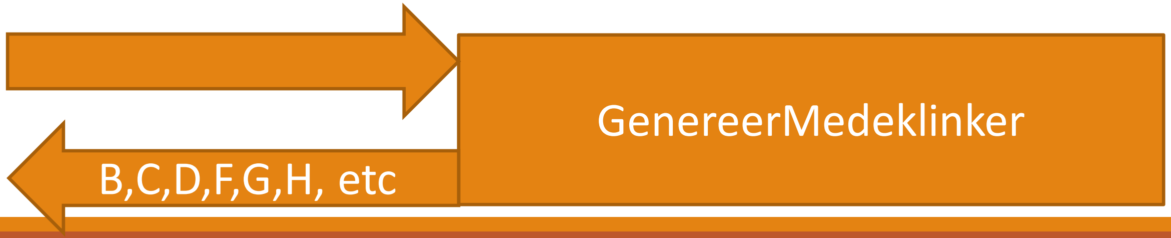
```
char GenereerKlinker()
```


GenereerKlinker

E,I,U,O of A

Methode die enkel medeklinkers genereert:

```
char GenereerMedeklinker()
```


GenereerMedeklinker

B,C,D,F,G,H, etc

IsKlinker

```
bool IsKlinker(char teken)
{
    switch (teken)
    {
        case 'E':
        case 'I':
        case 'O':
        case 'A':
        case 'U':
            return true;
        default:
            return false;

    }

}
```

GenereerKlinker

```
char GenereerKlinker()
{

    int waarde = r.Next(0, 5);
    switch (waarde)
    {
        case 0: return 'E';
        case 1: return 'O';
        case 2: return 'I';
        case 3: return 'U';
        case 4: return 'A';
    }
    return ' ';
}
```

GenereerMedeKlinker

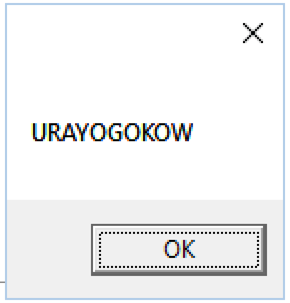
Zou ook met switch kunnen, maar we gaan nu eens (omdat het kan) deIsKlinkermethode herbruiken (en zo een pak minder code dan met een switch schrijven)

We genereren de hele tijd een letter (tussen A en Z) tot we er een hebben die géén Klinker is:

```
char GenereerMedeklinker()
{
    char result = 'E';
    while (IsKlinker(result))
    {
        int randomgetal = r.Next(65, 91);
        result = (char)randomgetal;
    }
    return result;
}
```

Trivia van de dag:Theoretisch gezien zou deze loop oneindige kunnen duren

Stap2: Naam generator, v2



URAYOGOKOW

```
string GenerateNameBetter(int namelength=6)
{

    string name = "";
    char vorigteken = (char)r.Next(65, 91);
    for (int i = 0; i < namelength; i++)
    {
        if (IsKlinker(vorigteken))
            vorigteken = GenereerMedeklinker();
        else vorigteken = GenereerKlinker();
        name += vorigteken;

    }
    return name;
}
```

Optionele parameter

ZinGenerator

ZinGenerator

Een eenvoudige zin kan bestaan uit:

- Onderwerp
- Werkwoord
- Lijdend voorwerp

Voorbeeld:

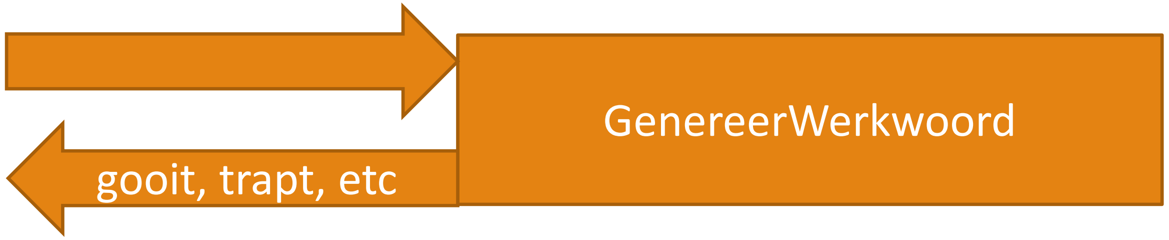
- Tim gooit de bal

3 Generators nodig

Onderwerp => Kunnen we NaamGenerator voor gebruiken

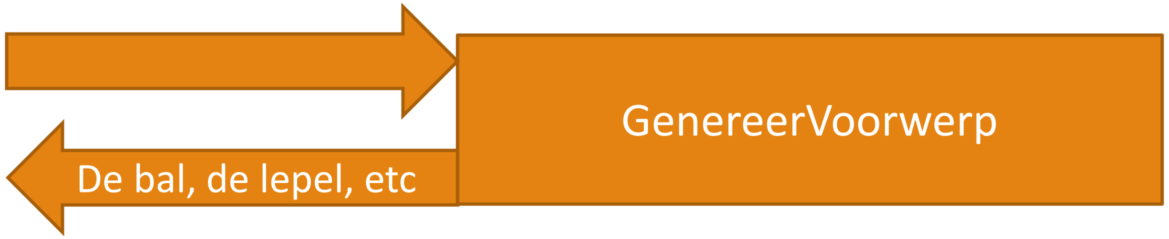
Werkwoord :

```
string GenereerWerkwoord()
```



Lijdend voorwerp:

```
string GenereerVoorwerp()
```



Werkwoord- en Voorwerpgenerator

Werkwoord:

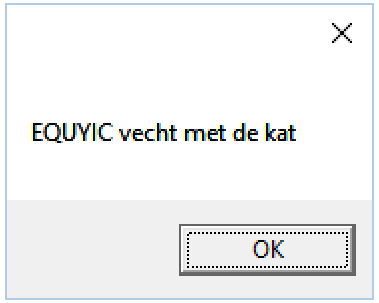
```
string GenereerWerkwoord()
 {
     switch (r.Next(0, 10))
     {
         case 0: return "roept";
         case 1: return "gooit";
         case 2: return "aait";
         case 3: return "eet";
         case 4: return "pakt";
         case 5: return "kijkt naar";
         case 6: return "ledigt";
         case 7: return "vecht met";
         case 8: return "beklimt";
         case 9: return "begraaft";
         default:
             return "IETS ONBEKENDS";
     }
 }
```

Voorwerp:

```
string GenereerVoorwerp()
{
    switch (r.Next(0, 10))
    {
        case 0: return "een bal";
        case 1: return "de hond";
        case 2: return "de kat";
        case 3: return "een lepel";
        case 4: return "het kind";
        case 5: return "het boek";
        case 6: return "de computer";
        case 7: return "een vork";
        case 8: return "het scherm";
        case 9: return "een dvd";
        default:
            return "IETS ONBEKENDS";
    }

}
```

ZinGenerator



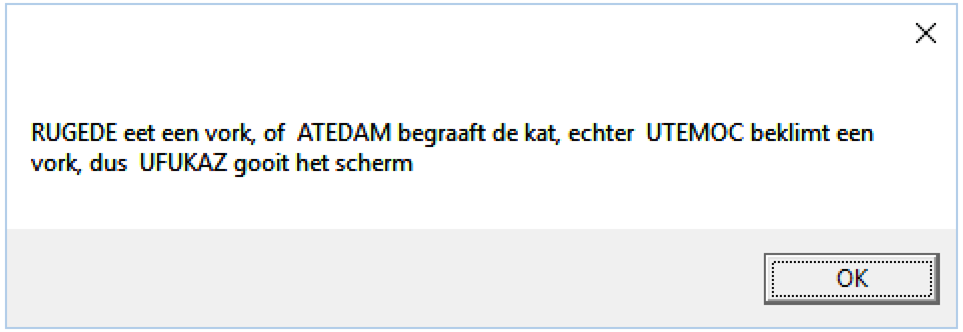
EQUYIC vecht met de kat

```
string GenereerKorteZin()
{
    string onderwerp = GenerateNameBetter(6);
    string werkwoord = GenereerWerkwoord();
    string lv = GenereerVoorwerp();

    string zin = onderwerp + " " + werkwoord + " " + lv;

    return zin;
}
```

Verhaalgenerator



RUGEDE eet een vork, of ATEDAM begraaft de kat, echter UTEMOC beklimt een vork, dus UFUKAZ gooit het scherm

Verhaalgenerator

We hebben nu genoeg kennis om langere zinnen en zelfs verhalen te generen.

Bijvoorbeeld:

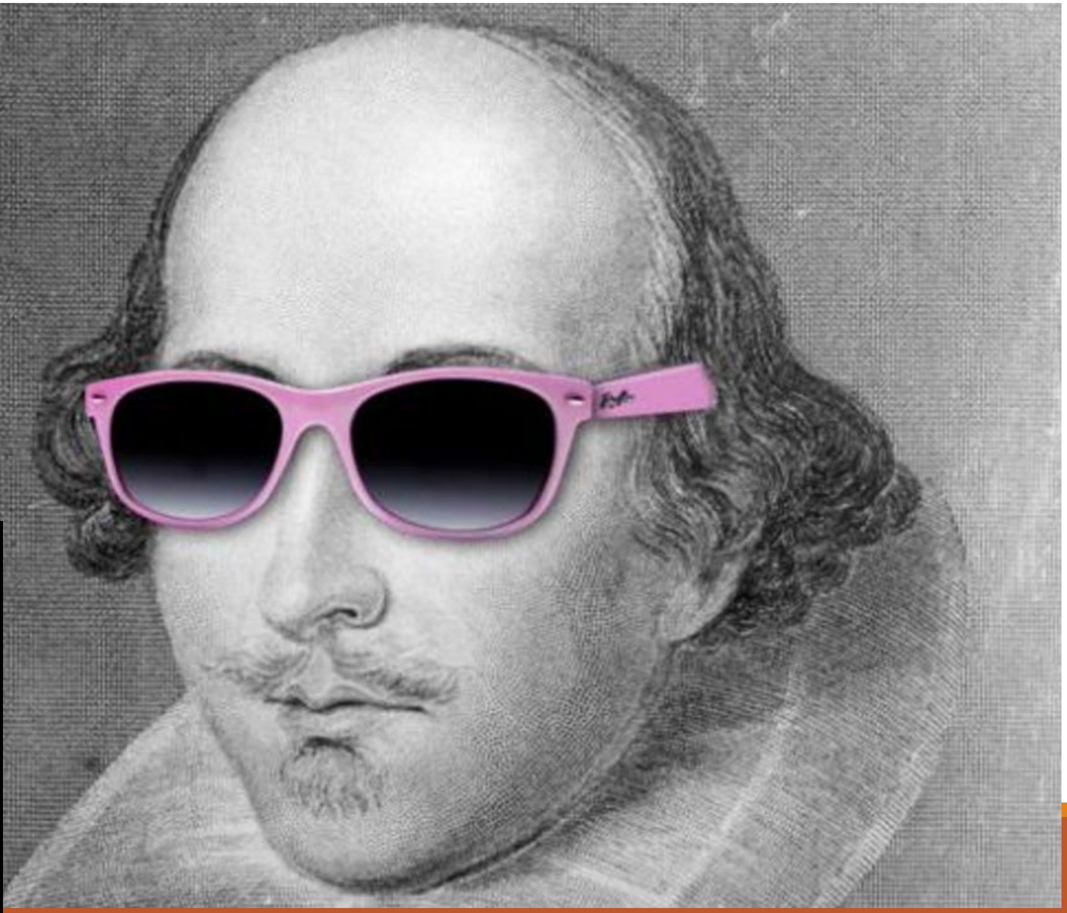
```
private string GenereerLangeZin(int bijzinlengte)
{
    string hoofdzin = GenereerKorteZin();
    for (int i = 0; i < bijzinlengte; i++)
    {
        hoofdzin += GenereerVoegwoord() + " " + GenereerKorteZin();
    }
    return hoofdzin;
}
```

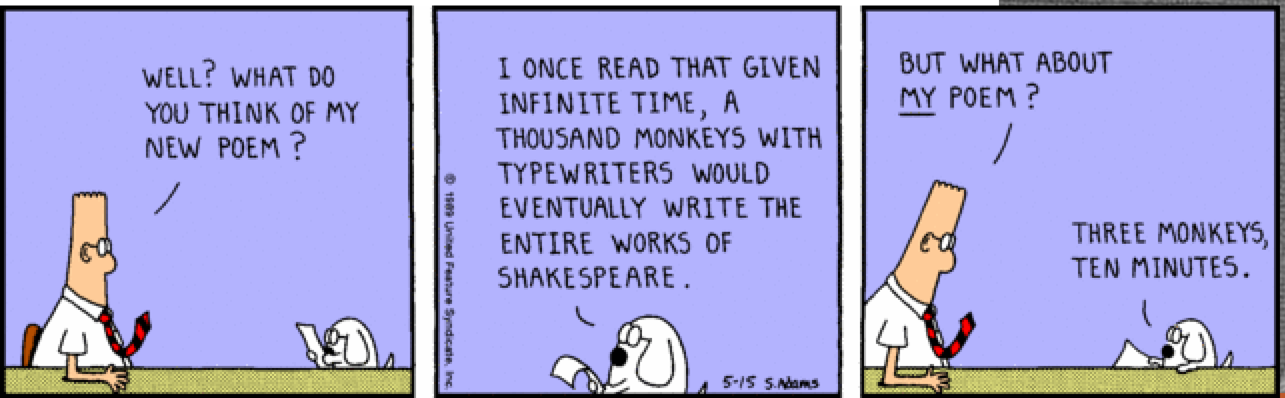
```
string GenereerVoegwoord()
       {
           switch (r.Next(0, 6))
           {
               case 0: return " en ";
               case 1: return ", maar ";
               case 2: return ", echter ";
               case 3: return ", dus ";
               case 4: return ", of ";
               case 5: return ", doch";

               default:
                   return "IETS ONBEKENDS";
           }
       }
```

Ben jij de volgende Shakespeare?! ☺

Maak zelf een methode “Genereerverhaal()”!




WELL? WHAT DO YOU THINK OF MY NEW POEM?
I ONCE READ THAT GIVEN INFINITE TIME, A THOUSAND MONKEYS WITH TYPEWRITERS WOULD EVENTUALLY WRITE THE ENTIRE WORKS OF SHAKESPEARE.
© 1989 United Feature Syndicate, Inc.
5-15 S.Adams
BUT WHAT ABOUT MY POEM?
THREE MONKEYS, TEN MINUTES.