Eindopdracht Vertalerbouw

VNVD

Studentassistent: J.S. Posthuma

S.L.C. Verberkt
s0166227
Calslaan 50-202
7522 MG Enschede

F.D. van Nee
s0166162
Nicolaas Maesstraat 63
7545 CD Enschede

13 juli 2009

Inhoudsopgave

1 Inleiding

Voor de eindopdracht van vertalerbouw was het de bedoeling om zelf een compiler voor een programmeertaal te ontwikkelen. De syntax van de programmeertaal mocht zelf worden gedefinieerd, maar de taal moest wel aan een aantal eisen voldoen. Het belangrijkste was, dat de taal een *expression language* moest zijn. Dit houdt in dat alles een expressie is, en dus alles een waarde op de stack achterlaat.

Voor het ontwikkelen van de compiler wordt gebruik gemaakt van ANTLR. ANTLR is een tool voor compiler ontwikkelaars die veel werk uit handen van de ontwikkelaar kan nemen. De compiler bestaat uit zes fases:

1. lexer; zet het input bestand om in een token stream (met ANTLR),
2. parser; analyseert de token stream en maakt er een Abstract Syntax Tree van (met ANTLR),
3. pre-checker; loopt over de namespaces en klassen in de AST heen om tijdens het checken klassen en members te kunnen resolven (handmatige tree walker die slechts over een klein deel loopt).
4. checker; loopt over de AST heen en bekijkt de context beperkingen, zoals type checking (ANTLR tree walker).
5. pre-code generator; loopt over de namespaces en klassen in de AST heen om de assembly headers, waarin klassen en members gedefinieerd worden, te definiëren (handmatige tree walker die slechts over een klein deel loopt).
6. code generator; loopt nogmaals over de AST heen en schrijft bytecode weg. Er is door ons gekozen om CIL code te genereren. Er worden dus .NET assemblies gegenereerd die kunnen draaien op de CLR of Mono (ANTLR tree walker).

De compiler zelf is geschreven in C#. ANTLR genereert C# code, en de extra programmatuur, zoals een symbol table en uitgebreide AST nodes is ook ontwikkeld in C#.

2 Beschrijving

De programmeertaal VNVD, de Van Nee-Verberkt-Duijvestijn taal, is een objectgeorienteerde taal in de stijl van C# en C++. Echter, het grote verschil met die talen is dat zij ook statements bevatten; VNVD kent louter expressies. Alles laat een waarde achter op de stack. Er is vrij exact vastgehouden aan de *Basic Expression Language*, beschreven in het dictaat. Hiernaast kent VNVD nog een groot aantal features, die gezamenlijk de taal *bijna* net zo krachtig maken als C# of C++.

VNVD is volledig compatibel met bestaande .NET framework klassen. Daardoor kan alle functionaliteit die in dit framework zit, of door anderen aan libraries voor dit framework is gemaakt, direct in VNVD worden gebruikt.

Als een volledig object georiënteerde taal kent VNVD ook een grote hoeveelheid concepten uit de object georiënteerde theorie. Zo kent de taal namespaces, waardoor code makkelijk te structureren is. Externe namespaces kunnen ook op bestands niveau geïmporteerd worden om gebruikt te worden.

Logischer wijs kent de taal klassen, maar ook interfaces en abstracte klassen. Voor klassen kent zij single inheritance en voor interfaces (en abstrace klassen) gewone inheritance. Klassen kunnen methoden, velden en events herbergen. Overigens kunnen deze elementen natuurlijk ook statisch gedefiniëerd worden.

In VNVD is het ook mogelijk enumeraties te definiëren. Binnen VNVD vallen deze onder de object georiënteerde beleving.

Zo als in elke echte programmeer taal bezit VNVD ook een aantal belangrijke conditionele constructies. Een if else constructie (ofwel de ternary), een while loop en zelfs een for each loop zitten standaard in de taal. De laatst genoemde maakt het mogelijk om over een enumereerbaar object te itereren.

De taal kent een try catch finally constructie voor foutafhandeling. Het is in VNVD mogelijk excepties te genereren en naar behoeft met deze constructie weer af te vangen en af te handelen. Zoals de overige onderdelen van de taal, is ook het foutafhandelingssysteem volledig compatibel met .NET.

Alle objecten en overige waarden kunnen worden opgeslagen in variabelen. VNVD kent naast gewone variabelen ook arraytypen. Elk willekeurig type kan ook als arraytype gebruikt worden. Net zoals gewone variabelen kunnen arrays ook bij de declaratie direct gevuld (gedenoteerd) worden.

Alle variabele typen in VNVD zijn ook constant te gebruiken. Constanten worden direct door de compiler afgehandeld en in de gegenereerde code vervangen. Daardoor kunnen constanten enkel van primitieve typen zijn.

Tot slot kent VNVD, als expressie taal, natuurlijk een breed scala aan expressies. Naast de verwachte binaire operatoren, unaire operatoren, arithmetische operatoren en vergelijkings operatoren kent VNVD ook uitgebreidere operatoren. Zoals de eerder genoemde ternary en assignment expressies.

Vanwege de natuur van de taal geldt het lezen variabele (of velden) en de invocatie van methoden ook als een expressie. Daarnaast kent de taal twee speciale functies, namelijk de read() en de write() instructie die onderdeel zijn van de *Basic Expression Language*.

Ook kunnen alle waarden naar hartelust gecast worden. Hierbij gelden natuurlijk dezelfde regels in elke andere taal. Een cast blijft immers niet zonder risico.

Naast al deze constructies kent VNVD nog een aantal andere handige concepten die het programmeren vergemakkelijken en de taal zelf krachtiger maken.

3 Problemen en oplossingen

Bij het ontwerpen en implementeren van de compiler voor onze taal zijn ook wij tegen problemen aangelopen. Er moesten in een vroeg stadium al verschillende ontwerpkeuzes gemaakt worden die later, in een verder stadium van het ontwerpproces, soms verkeerd uit bleken te pakken. Desalniettemin is voor de meeste problemen die we zijn tegengekomen een elegante, nette oplossing gevonden.

Dat gezegd hebbende merken wij op dat het lastigste onderdeel van de compiler zonder twijfel de checker is. Een checker voor een willekeurige objectgeoriënteerde programmeertaal moet sowieso ontzettend gecompliceerd zijn. Één van de redenen hiervoor is het feit dat zulke talen de mogelijkheid tot inheritance hebben. Types van expressies hoeven vaak niet exact gelijk te zijn, als de één maar een superklasse of interface is van de ander. Dit verschijnsel wordt ook wel polymorphism genoemd en is één van de bouwstenen van objectgeoriënteerde talen.

Laatstgenoemde en veel andere problemen leverde interessante discussiepunten op voor de implementatie van onze taal. In de volgende paragrafen zullen we uitleggen wat voor ons de meeste problemen opleverde en de door ons gekozen oplossing voor die problemen. We splitsen dit op in drie delen, namelijk de problemen bij het maken van de parser, de checker en de generator. Het specificeren van de lexer leverde geen noemenswaardige problemen op en is daarom dus weggelaten uit dit hoofdstuk.

3.1 Parser

Het lastige aan het schrijven van de parser was voornamelijk dat de taal LL(1) moest zijn. Het was nodig om systematisch de grammatica op te stellen zodat left-recursion en beslissingen die niet LL(1) waren te vermijden. Hoewel een deel dus opgelost is door het herschrijven van de grammar, is een ander deel opgelost met behulp van syntactic predicates.

We beschrijven nu één probleem waarvoor het veel tijd heeft gekost om een goede oplossing te vinden.

Klassen, methoden en velden hebben allen modifiers bij hun declaratie. Deze specificeren de zichtbaarheid, of dat een methode static is of dat deze een methode uit een superklasse overschrijft. In onze oorspronkelijke specificatie van de taal was in de parser niet gespecificeerd welke modifier in welke volgorde waar mocht staan, maar was er een algemene regel die alle modifiers toeliet. In een definitie van een methode werd deze regel dan gebruikt zoals hieronder beschreven.

```
1  modifier : PRIVATE | PUBLIC | STATIC
2       | ABSTRACT | ...;
3  method_decl : modifier* type_qualifier
4       IDENTIFIER LPAREN parameter_list RPAREN
5       block_expression;
```

Deze regel was in deze vorm echter nooit LL(1) te krijgen, omdat het aan het begin 0 of meer modifiers kon matchen. Eerst was gekozen om een keyword 'method' te gebruiken die aan het begin van de method definitie moest staan. Dit loste de problemen op, maar was niet netjes. Daarom is later gekozen om de regel modifier op te splitsen in meerdere andere regels, waardoor het niet meer

nodig is om 0 of meer te matchen. Bijvoorbeeld, de methodedeclaratie ziet er nu als volgt uit:

```
1  method_decl : method_mod_list type_qualifier
2      IDENTIFIER LPAREN parameter_list RPAREN
3      block_expression;
4  method_mod_list : access_modifier
5      (STATIC | virtual_modifier)?;
6  access_modifier : PRIVATE | PUBLIC
7      | PROTECTED | INTERNAL;
8  virtual_modifier : VIRTUAL | OVERRIDE;
```

Op deze manier is het mogelijk om de taal zowel netjes en overzichtelijk als ook LL(1) te houden.

3.2 Checker

In de checker traden ook verschillende problemen op. Zo moest ervoor gezorgd worden dat eerst alle klassen gedefinieerd werden, vervolgens de methoden en daarna pas de inhoud van de methoden. Het moet in deze volgorde, omdat de een afhankelijk is van de ander. Voor de definitie van methoden moet je weten welke klassen er zijn. Immers, een return type van een methode kan een zelfgedefinieerde klasse zijn. Er moet wel bekend zijn op dat moment of die klasse ook echt bestaat.

Er traden ook problemen op bij het maken van compatibiliteit van .NET Framework klassen. Ons eerste idee was om de symbol table uit te breiden met functionaliteit voor named scopes, om zo de mogelijkheid te hebben om klassen en methoden hierin op te slaan. Later is van dit idee afgeweken, omdat zo bestaande .NET klassen en zelfgedefinieerde klassen op verschillende manieren werden aangesproken.

Er is toen naar een oplossing gezocht waarbij er geen verschil zou zijn tussen het zoeken naar bestaande .NET klassen en zelfgedefinieerde klassen. Deze oplossing is gevonden door de bestaande .NET Reflection klassen te extenden. In onze oplossing wordt bijvoorbeeld een klasse of interface gerepresenteerd door de abstracte klasse Type, waarbij een standaard .NET klasse een RuntimeType is en een zelfgedefinieerde klasse een UserType. Iets soortgelijks geldt voor methoden, velden en dergelijke, waardoor alles volgens dezelfde interface aangesproken kan worden.

3.3 Code generator

Wanneer een checker goed is ontworpen levert het maken van de generator vaak nauwelijks problemen op. De code templates van de specificatie van een taal kunnen bijna letterlijk worden overgenomen bij het maken van de generator.

Een aantal punten leverden echter nog problemen op, mede doordat er toch een aantal punten in de checker waren die beter ontworpen hadden kunnen worden. Het belangrijkste punt hiervan is dat de checker trees niet meer herschrijft. Op bepaalde punten, met name bij methode/veld aanroepen, maar zeker ook bij bepaalde expressies en constanten was het handiger geweest om in de checker tree rewrite toe te passen. Er was echter in een vroeg stadium voor gekozen om dit niet te doen en dit was niet meer makkelijk te veranderen.

Ook het implementeren van meerdere assignments leverde problemen op. Neem bijvoorbeeld de assignment *this::a = this::b = 5 * c*. Zowel this::b en

this::a krijgen hier de waarde toegekend van *5 * c*. Nu zijn er twee mogelijkheden om dit te realiseren:

- door de expressie *5 * c* tweemaal uit te voeren, één per assignment, of,
- door de expressie *5 * c* eenmaal uit te voeren, deze in een tijdelijke variabele op te slaan en vervolgens dit gebruiken om de assignment uit te voeren.

Het is niet mogelijk om de expressie uit te voeren gevolgd door de duplicate opcode, omdat de linkerkant van een assignment in onze taal ook kan bestaan uit een instantievariabele, waarbij een referentie naar de this pointer dus al op de stack staat. Een assignment na een duplicate uitvoeren zou dan niet werken, vanwege verkeerde waardes op de stack.

Wij hebben gekozen voor de tweede oplossing waarbij een tijdelijke variabele gebruikt wordt. Hiervoor hebben wij gekozen, omdat het nutteloos is dezelfde expressie tweemaal uit te voeren. Dit is alleen maar inefficiént.

4 Taalbeschrijving

Deze sectie bevat de volledige beschrijving van VNVD. Deze beschrijving bestaat uit de syntax, de context beperkingen en de semantische regels.

4.1 Programmastructuur

Onde programmastructuur valle alle regels die de basis structuur van een programma definiëren. Dit zijn de regels voor namespaces en import statements. Alle leden van de namespaces vallen onder de klassenstructuur.

4.1.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle programmastructuur gerelateerde regels.

```
1  program
2      ::= import_stat* namespace_decl*
3
4  import_stat
5      ::= USING qualifier SEMICOLON
6
7  namespace_decl
8      ::= NAMESPACE qualifier LCURLYBRACE namespace_body RCURLYBRACE
9
10 namespace_body
11     ::= (class_decl | interface_decl | enum_decl)*
```

4.1.2 Context-beperkingen

De programmastructuur kent geen context-beperkingen.

4.1.3 Semantiek

Dit zijn de semantische regels voor de programmastructuur.

- Een programma 'P' wordt gedraaid door de Main methode, gedefinieerd in één van zijn klassen uit te voeren. Wanneer er geen Main methode is, is het programma een library.

4.2 Klassenstructuur

Dit zijn de regels met betrekking tot de klassen, interfaces en enumeraties.

4.2.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle klassenstructuur gerelateerde regels.

```
1  class_mod_list
2      ::= access_modifier (ABSTRACT)?
3
4  class_decl
5      ::= CLASS class_mod_list qualifier (EXTENDS qualifier)? (
           IMPLEMENTS type_list)? class_body
6
```

```
7  class_body
8      ::= LCURLYBRACE class_element* RCURLYBRACE
9
10 enum_decl
11     ::= ENUM access_modifier qualifier enum_body
12
13 interface_decl
14     ::= INTERFACE class_mod_list qualifier (EXTENDS qualifier)?
           interface_body
15
16 interface_body
17     ::= LCURLYBRACE interface_method* RCURLYBRACE
```

4.2.2 Context-beperkingen

Hier volgen de context-beperkingen voor de klassenstructuur.

- Er mogen geen twee types met dezelfde naam gedefinieerd worden.
- Er mogen geen twee members van klassen zijn met dezelfde naam en parameters.
- De *qualifier* bij de regel *EXTENDS* bij een klassendeclaratie moet een andere bestaande klasse zijn.
- De lijst met *qualifier*s bij de regel *IMPLEMENTS* bij een klassendeclaratie moeten allen andere bestaande interfaces zijn.
- De qualifier bij de regel *EXTENDS* bij een interfacedeclaratie moet een bestaande interface zijn.
- Wanneer een interface geïmplementeerd wordt door een klasse moet deze klasse alle methoden van de interface implementeren.
- Wanneer een klasse een abstracte klasse uitbreid, moeten alle abstracte methoden geïmplementeerd worden.

4.2.3 Semantiek

De klassenstructuur kent de volgende semantische regels:

- Een klassendeclaratie zorgt ervoor dat er een klasse in de huidige namespace wordt gedefineerd met de gegeven naam.
- Een enumeratiedeclaratie zorgt ervoor dat er een enumeratie in de huidige namespace wordt gedefineerd met de gegeven naam.
- Een interfacedeclaratie zorgt ervoor dat er een interface in de huidige namespace wordt gedefineerd met de gegeven naam.
- Als een klasse of interface iets anders extend, dan betekent dat dat de members van de base class ook beschikbaar zijn in de hier gedefinieerde klasse.

4.3 Klassenleden

Dit gedeelte handelt over de leden van klassen, interfaces en enumeraties. Dit kan onder andere een methode of een veld zijn.

4.3.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle klassenleden gerelateerde regels.

```
1  enum_body
2      ::= LCURLYBRACE (IDENTIFIER (COMMA IDENTIFIER)*)? RCURLYBRACE
3
4  interface_method
5      ::= type_qualifier IDENTIFIER LPAREN parameter_list RPAREN
           SEMICOLON
6
7  method_mod_list
8      ::= access_modifier (STATIC | virtual_modifier)?
9
10 field_mod_list
11     ::= access_modifier STATIC? INITONLY?
12
13 constructor_mod_list
14     ::= access_modifier
15
16 abstract_mod_list
17     ::= access_modifier ABSTRACT
18
19 type_list
20     ::= qualifier (COMMA qualifier)*
21
22 class_element
23     ::= method_decl
24     |   constructor_decl
25     |   abstract_method_decl
26     |   field_decl
27     |   static_constructor
28
29 static_constructor
30     ::= STATIC block_expression
31
32 field_decl
33     ::= field_mod_list qualifier IDENTIFIER (COMMA IDENTIFIER)*
           SEMICOLON
34
35 method_decl
36     ::= method_mod_list type_qualifier IDENTIFIER LPAREN
           parameter_list RPAREN block_expression
37
38 abstract_method_decl
39     ::= abstract_mod_list type_qualifier IDENTIFIER LPAREN
           parameter_list RPAREN SEMICOLON
40
41 constructor_decl
42     ::= constructor_mod_list IDENTIFIER LPAREN parameter_list
           RPAREN (COLON BASE LPAREN argument_list RPAREN)?
           block_expression
43
44 parameter_list
45     ::= (qualifier IDENTIFIER (COMMA qualifier IDENTIFIER)*)?
46
```

```
47  access_modifier
48      ::= PRIVATE | PUBLIC | PROTECTED | INTERNAL
49
50  virtual_modifier
51      ::= VIRTUAL | OVERRIDE
```

4.3.2 Context-beperkingen

De context-beperkingen van de klassenleden zijn als volgt:

- Bij een enumeratie moeten alle *IDENTIFIER*s verschillend zijn.
- Alle type_qualifiers moeten bestaande types zijn die beschikbaar zijn met de gebruikte imports.
- Een klasse member mag alleen als *ABSTRACT* gedefinieerd worden wanneer de klasse ook *ABSTRACT* is.
- De modifier *OVERRIDE* mag alleen gebruikt worden wanneer een methode gedefinieerd wordt die in een superklasse als *VIRTUAL* staat gemarkeerd.
- Er mag per klasse maar één *STATIC* constructor zijn.

4.3.3 Semantiek

De semantische regels voor de klassenleden zijn als volgt:

- De modifier *STATIC* betekent dat deze member zonder instantie van de klasse aanroepbaar is
- De modifier *OVERRIDE* betekent dat deze methode een member van een superklasse overschrijft
- De modifier *VIRTUAL* betekent dat deze methode overschrijfbaar is voor subklassen
- De modifier *ABSTRACT* betekent dat deze methode in een subklasse geïmplementeerd moet worden
- De modifier *INITONLY* betekent dat dit veld enkel in een van de constructoren geïnitializeerd mag worden. Daarna is dit niet meer mogelijk.
- De modifier *PRIVATE* betekent dat deze member alleen binnen de klasse zelf beschikbaar is
- De modifier *PUBLIC* betekent dat deze member overal beschikbaar is
- De modifier *PROTECTED* betekent dat deze member beschikbaar is in zowel de klasse zelf als subklassen
- De modifier *INTERNAL* betekent dat deze member overal in de assembly waarin hij is gedefinieerd beschikbaar is
- Het definiëren van members heeft als gevolg dat deze methoden, fields of enumeraties op hun klasse aanroepbaar zijn.
- Fields en methoden returnen een bepaald type, zoals hun *type_qualifier* aangeeft.

4.4 Statements

In deze sectie worden de syntax, context-beperkingen en semantiek van statements uitgebreid toegelicht.

4.4.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle statement gerelateerde regels.

```
1  statement
2      ::= declaration_statement SEMICOLON
3      |   const_decl_statement SEMICOLON
4      |   expression SEMICOLON
5      |   SEMICOLON
6      |   control_statement
7
8  control_statement
9      ::= while_statement
10     |   for_statement
11     |   try_statement
12     |   throw_statement
13
14 throw_statement
15     ::= THROW expression SEMICOLON
16
17 try_statement
18     ::= TRY block_expression (finally_block | (catch_block+
           finally_block?))
19
20 catch_block
21     ::= (CATCH | CAREBOX) LPAREN qualifier IDENTIFIER RPAREN
           block_expression
22
23 finally_block
24     ::= FINALLY block_expression
25
26 while_statement
27     ::= WHILE LPAREN scope_expr RPAREN block_expression
28
29 for_statement
30     ::= FOR LPAREN qualifier IDENTIFIER IN expression RPAREN
           block_expression
```

4.4.2 Context-beperkingen

De context-beperkingen van de statements zijn als volgt:

- De expressie van een *throw_statement* moet een type opleveren dat van *System.Exception* afstamt.
- De *qualifier* die in een *catch_block* staat moet van een type zijn dat van *System.Exception* afstamt.
- De *scope_expr* van een *while_statement* moet een *System.Boolean* type opleveren.
- De expression van een *for_statement* moet een type opleveren die de interface *System.Collections.IEnumerable* implementeert.

4.4.3 Semantiek

De semantische regels van de statements zijn als volgt:

- Het *throw_statement* onderbreekt de programmauitvoering een gooit de exceptie die de expression oplevert.
- Het *try_statement* begint een block die excepties kan afvangen. Excepties gespecificeerd een de catch blokken worden afgevangen.
- Het *catch_block* specificeert de exceptie die wordt afgevangen. Excepties die van het type van *qualifier* zijn, of van een subklasse hiervan worden afgevangen. Dit block wordt alleen uitgevoerd wanneer zo'n exceptie wordt gegooid.
- Het *try_block* wordt altijd uitgevoerd, ook als er een exceptie is gegooid in het try gedeelte.
- Declaraties gedaan in de *scope_expr* van een *while_statement* blijven het gehele while statement in scope.
- De *while_statement* wordt net zolang uitgevoerd tot de scope_expr false oplevert.
- De *for_statement* itereert over alle object heen die de expression van *System.Collections.IEnumerable* opleveren. De variabele met de naam van *IDENTIFIER* wordt steeds geupdate met een van de object uit de lijst met object waar overheen geïtereerd wordt.

4.5 Declaraties

In deze sectie worden declaraties, hun context-beperkingen en semantiek besproken.

4.5.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle declaratie gerelateerde regels.

```
1  declaration_statement
2      ::= qualifier IDENTIFIER BECOMES expression
3      |   qualifier IDENTIFIER (COMMA IDENTIFIER)*
4
5  const_decl_statement
6      ::= CONST qualifier IDENTIFIER BECOMES const_value (COMMA
           IDENTIFIER BECOMES const_value)*
7
8  const_value
9      ::= literal_value
10     |   LCURLYBRACE literal_value (COMMA literal_value)*
           RCURLYBRACE
```

4.5.2 Context-beperkingen

De context-beperkingen van de declaraties zijn als volgt:

- Bij een declaratie mag het niet zo zijn dat er in dezelfde scope al een andere declaratie met dezelfde *IDENTIFIER* bestaat.
- Wanneer een declaratie van een variabele of constante gelijk geïnitializeerd wordt moet de *expression* van hetzelfde type zijn als het type van de variabele die gedeclareerd wordt.

4.5.3 Semantiek

De semantische regels van de declaraties zijn als volgt:

- De declaratie van een variabele zorgt ervoor dat deze variabele later kan worden gebruikt in expressies. De variabele wordt van het gespecificeerde type.
- De declaratie van een constante zorgt ervoor dat deze constante in expressies gebruikt kan worden. De waarde van deze constante wordt direct gebruikt in de code.
- Wanneer de constante een array type is kan in het vervolg alleen aan elementen van die array gerefereerd worden.

4.6 Expressies

Gezien VNVD een expressie taal is, kent de taal vele expressies. Een expressie is een operatie die een waarde achter laat op de stack van de virtuele machine.

4.6.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle expressie gerelateerde regels.

```
1  expression
2      ::= assignment_expression
3
4  scope_expr
5      ::= statement+
6
7  assignment_expression
8      ::= ternary_expression BECOMES assignment_expression
9      |   ternary_expression (ADDEVENT | REMEVENT)
           assignment_expression
10     |   ternary_expression
11
12 object_creation_expression
13     ::= NEW qualifier LPAREN argument_list RPAREN
14
15 array_creation_expression_elemented
16     ::= NEW qualifier LCURLYBRACE expression (COMMA expression)*
           RCURLYBRACE
17
18 array_creation_expression
19     ::= NEW qualifier LBRACKET expression RBRACKET
```

```
20
21  ternary_expression
22      ::= IF LPAREN scope_expr RPAREN THEN expression (ELSE
            expression)? FI
23      |   conditional_or_operator_expression
24
25  conditional_or_operator_expression
26      ::= conditional_and_operator_expression (OR
            conditional_and_operator_expression)*
27
28  conditional_and_operator_expression
29      ::= logic_or_operator_expression (AND
            logic_or_operator_expression)*
30
31  logic_or_operator_expression
32      ::= logic_xor_operator_expression (LOR
            logic_xor_operator_expression)*
33
34  logic_xor_operator_expression
35      ::= logic_and_operator_expression (LXOR
            logic_and_operator_expression)*
36
37  logic_and_operator_expression
38      ::= logic_expression (LAND logic_expression)*
39
40  logic_expression
41      ::= cast_as_expression ((LE | LEQ | GE | GEQ | EQ | NEQ)
            cast_as_expression)*
42
43  cast_as_expression
44      ::= primary_expression ((AS | IS) qualifier)*
45
46  primary_expression
47      ::= secundary_expression ((PLUS | MINUS) secundary_expression)*
48
49  secundary_expression
50      ::= logicnotoperator_expression ((MULTIPLY | DIVISION | MOD)
            logicnotoperator_expression )*
51
52  logicnotoperator_expression
53      ::= (NOT | PLUS | MINUS)* cast_expression
54
55  cast_expression
56      ::= (LPAREN qualifier RPAREN cast_expression)
57      |   invocation_or_load_expression
58
59  invocation_or_load_expression
60      ::= operand ((DCOLON IDENTIFIER (LPAREN argument_list RPAREN)?)
             | (LBRACKET expression RBRACKET))*
61
62  argument_list
63      ::= (expression (COMMA expression)*)?
64
65  operand
66      ::= qualifier
67      |   literal_value
68      |   THIS
69      |   BASE
70      |   object_creation_expression
71      |   array_creation_expression
72      |   array_creation_expression_elemented
73      |   READ LPAREN IDENTIFIER RPAREN
```

```
74        |    READ LPAREN IDENTIFIER (COMMA IDENTIFIER)+ RPAREN
75        |    WRITE LPAREN expression RPAREN
76        |    WRITE LPAREN expression (COMMA expression)+ RPAREN
77        |    LPAREN expression RPAREN
78        |    block_expression
79
80  literal_value
81        ::= NUMBER
82        |    FLOAT
83        |    STRING
84        |    CHAR
85        |    TRUE
86        |    FALSE
87        |    NULL
88
89  block_expression
90        ::= LCURLYBRACE statement* RCURLYBRACE
```

4.6.2 Context-beperkingen

De context-beperkingen voor de expressie regels zijn als volgt:

- In het geval van een *assignment_expression* zijn er twee mogelijke context beperkingen. Indien de regel een *BECOMES* token bevat dienen beide zijden van hetzelfde type te zijn. De linker zijde moet dan ook assignable zijn. In het geval dat er een *ADDEVENT* of *REMEVENT* staat dient de linker zijde een event te zijn en de rechterzijde van het type *System.EventHandler* te zijn.

- In een *object_creation_expression* dient de *qualifier* de naam van een bestaande klasse te zijn. Daarnaast dient de *argument_list* ofwel leeg, ofwel passend te zijn op de parameters van een bestaande constructor van de klasse waarnaar de *qualifier* wijst.

- De *qualifier* in een *array_creation_expression_elemented* dient te verwijzen naar een bestaand array type. Daarnaast dienen de typen van alle *expression*s gelijk te zijn aan het enkele type van de *qualifier* die vooraan staat.

- De *qualifier* in een *array_creation_expression* dient te verwijzen naar een bestaand type. Daarnaast dient de *expression* van het type *System.Int32* te zijn en daarnaast ook nog een positief getal te zijn.

- Een *ternary_expression* vereist dat de gebruikte *scope_expr* van het type *System.Boolean* is.

- Voor de *conditional_and_operator_expression* gelden dezelfde beperkingen als voor de *conditional_or_operator_expression*, namelijk dat beide zijden van de expressie van het type *System.Boolean* dienen te zijn. Deze context beperking geldt echter niet als in de regel de tokens *AND* en *OR* niet voor komen.

- Voor de *logic_and_operator_expression*, de *logic_or_operator_expression* en de *logic_xor_operator_expression* geldt dat beide zijden van de expressie

van hetzelfde type dienen te zijn. Daarnaast dienen beiden van een primitief type te zijn of een type dat, afhankelijk van de gebruikte tokens de volgende methode kent:

LAND static <Type> op_BitwiseAnd(<Type>, <Type>)

LOR static <Type> op_BitwiseOr(<Type>, <Type>)

LXOR static <Type> op_ExclusiveOr(<Type>, <Type>)

Indien er geen token gebruikt wordt in de regel geldt deze beperking niet.

- De beide zijden van een *logic_expression* dienen van hetzelfde type te zijn. Daarnaast dienen beiden van een primitief type te zijn of een type dat, afhankelijk van de gebruikte tokens de volgende methode kent:

  LE static <Type> op_LessThan(<Type>, <Type>)

  LEQ static <Type> op_LessThanOrEqual(<Type>, <Type>)

  GE static <Type> op_GreaterThan(<Type>, <Type>)

  GEQ static <Type> op_GreaterThanOrEqual(<Type>, <Type>)

  EQ static <Type> op_Equality(<Type>, <Type>)

  NEQ static <Type> op_Inequality(<Type>, <Type>)

  Indien er geen token gebruikt wordt in de regel geldt deze beperking niet.

- De *qualifier* in de *cast_as_expression* dient te verwijzen naar dezelfde klasse als, een superklasse van of een interface geïmplementeerd door het type van de linker zijde van de expressie.

- Beide zijden in een *primary_expression* dienen van hetzelfde type te zijn, indien de regel het token *PLUS* of het token *MINUS* bevat. In dat geval dienen beide typen ook ofwel primair ofwel van een type dat de bijbehorende functie implementeerd te zijn:

  PLUS static <Type> op_Addition(<Type>, <Type>)

  MINUS static <Type> op_Substraction(<Type>, <Type>)

- De beide zijden van een *secundary_expression* dienen van hetzelfde type te zijn. Daarnaast dienen beiden van een primitief type te zijn of een type dat, afhankelijk van de gebruikte tokens de volgende methode kent:

  MULTIPLY static <Type> op_Multiply(<Type>, <Type>)

  DIVISION static <Type> op_Division(<Type>, <Type>)

  MOD static <Type> op_Modulus(<Type>, <Type>)

  Indien er geen token gebruikt wordt in de regel geldt deze beperking niet.

- De gebruikte *cast_expression* in een *logicnotoperator_expression* dient van een primitief type te zijn of van een type dat, afhankelijk van de gebruikte tokens de volgende methoden kent:

  NOT static <Type> op_LogicalNot(<Type>, <Type>)

  PLUS static <Type> op_UnaryPlus(<Type>, <Type>)

MINUS static <Type> op_UnaryNegation(<Type>, <Type>)

Indien er geen token gebruikt wordt in de regel geldt deze beperking niet.

- De *qualifier* in de *cast_expression* dient te verwijzen naar dezelfde klasse als, een superklasse van of een interface geïmplementeerd door het type van de rechter zijde van de expressie.

- De regel *invocation_or_load_expression* kent twee verschillende constructies met hun eigen context-beperkingen. Voor het gebruik van de token *DCOLON* geldt dat de *IDENTIFIER* moet refereren naar een methode van het type van de *operand*, daarnaast moeten de parameters van deze methoden ook te vullen zijn op basis van de argumenten uit de *argument_list*. Indien de token *LBRACKET* wordt tegen gekomen moet de *operand* van een array type zijn. Daarnaast dient de *expression* dan een positieve *System.Int32* terug te geven.

- Een *argument_list* wordt enkel in invocaties gebruikt. Deze dient dan overeen te komen met de parameters van de methode (of constructor) die aangeroepen wordt. In dit geval betekent dit de *argument_list* even veel *expression*s kent als de *parameter_list IDENTIFIER*s. Daarnaast dienen de typen van de, respectievelijke, *expression*s gelijk te zijn aan, of te verwijzen naar superklassen danwel interfaces van, de aldaar door de *qualifier*s aangewezen typen.

- In de *operand* regels geldt dat alle *IDENTIFIER*s gebruikt bij een *READ* van het type *System.Int32*, *System.Char* of *System.String* dienen te zijn.

4.6.3 Semantiek

De semantische regels zijn als volgt:

- Een *assignment_expression* waarin de token *BECOMES* gebruikt wordt wijst de waarde van de rechter zijde toe aan de variabele of het veld gespecificeerd door de linker zijde. Daarnaast retourneert de expressie de waarde van de rechter zijde. In het geval van een *ADDEVENT* wordt de event handler aan de rechter zijde gekoppeld aan het links gespecificeerde event. Indien het een *REMEVENT* betreft wordt de event handler juist verwijderd. Beiden retourneren niets.

- Een *object_creation_expression* maakt een nieuw object aan van het door de *qualifier* gespecificeerde type. Daarnaast wordt de constructor aangeroepen met de argumenten uit de *argument_list*. Het nieuwe object wordt vervolgens op de stack geplaatst.

- Een *array_creation_expression_elemented* maakt een array aan met als elementen de waarden uit de verscheidene *expression*s. Deze array wordt daarna terug gegeven.

- Een *array_creation_expression* creëert een array met een grootte gelijk aan de waarde van de *expression*. Daarna wordt de gemaakte array geretourneerd.

- De *ternary_expression* opent eerst een nieuwe scope. Vervolgens wordt de *scope_expr* geëvalueerd. Indien deze waar wordt bevonden wordt een nieuwe scope geopend en de eerste *expression* uitgevoerd. Daarna wordt de scope weer gesloten. Indien de *scope_expr* negatief wordt bevonden en er een *ELSE* bestaat wordt een nieuwe scope geopend en de tweede *expression* uitgevoerd en wederom de scope gesloten. Als deze er echter niet is wordt er niets gedaan. Tot slot wordt de laatste scope gesloten. Deze expressie geeft niets terug, behalve als beide *expression*s van hetzelfde type zijn. In dat geval blijft de waarde geretourneerde door de uitgevoerde *expression* op de stack staan.

- Een *conditional_or_operator_expression* wordt lazy geëvalueerd. Dit betekent dat eerst de linker zijde wordt geëvalueerd. Als deze waar oplevert wordt er waar terug gegeven. Anders wordt de rechter zijde geëvalueerd en de uitkomst daarvan terug gegeven.

- Een *conditional_and_operator_expression* wordt lazy geëvalueerd. Dit betekent dat eerst de linker zijde wordt geëvalueerd. Als deze onwaar oplevert wordt er onwaar terug gegeven. Anders wordt de rechter zijde geëvalueerd en de uitkomst daarvan terug gegeven.

- Een *logic_or_operator_expression* levert de waarde op van de logic or van zijn twee zijden.

- Een *logic_xor_operator_expression* levert de waarde op van de logic xor van zijn twee zijden.

- Een *logic_and_operator_expression* levert de waarde op van de logic and van zijn twee zijden.

- Een *logic_expression* levert de waarde op van zijn twee zijden vergeleken met elkaar. Dit gebeurt volgens de operator die daar gebruikt wordt. Dit is één van: *LE*, *LEQ*, *GE*, *GEQ*, *EQ* of *NEQ*.

- Een *cast_as_expression* verandert het type van de expressie waar hij op gebruikt wordt. Het type van deze expressie is gelijk aan het type van zijn *qualifier*. Wanneer het niet mogelijk is om te casten levert deze null op.

- Een *primary_expression* telt twee waarden bij elkaar op, of trekt ze van elkaar af. Het resultaat hiervan blijft achter op de stack.

- Een *secundary_expression* voert een vermenigvuldiging op twee waarden uit, deelt ze of neemt de modulus ervan. Het resultaat blijft achter op de stack.

- Een *logicnotoperator_expression* inverteert de waarde waar hij op uitgevoerd wordt, of levert hiervan de unaire plus of negatie op.

- Een *cast_expression* verandert het type van een *expression*. Het type van deze expressie is gelijk aan het type van zijn *qualifier*. Wanneer het niet mogelijk is om te casten wordt een exceptie gegooid.

- Een *invocation_or_load_expression* roept ofwel een methode aan, of laadt een veld. Het type van deze expressie is ofwel de return type van de methode, of die van het veld dat hij laadt. Een methode wordt aangeroepen met de parameters die meegegeven zijn. Deze worden eerst geëvalueerd.
- Een *literal_value* laadt een waarde van een literal op de stack.
- Het keyword *THIS* laadt de waarde van de klasse waarin hij zit op de stack.
- Het keyword *BASE* geeft aan dat methoden uit de superklasse van de klasse moeten worden aangeroepen. Dit wordt gebruikt bij invocaties.
- De *READ* functie leest een waarde van de standaardinvoer en plaatst deze in meegegeven identifier. Ook laat hij de waarde achter op de stack wanneer er slechts één parameter is.
- De *WRITE* functie schrijft een waarde naar de standaarduitvoer. Ook laat hij de waarde achter op de stack wanneer er slechts één parameter is.

4.7 Qualifiers

Qualifiers zijn volledige plaats aanduidingen naar typen. Een qualifier wordt dus gebruikt om de locatie van een klasse, enumeratie, interface of dergelijke te vinden.

4.7.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle qualifier gerelateerde regels.

```
1  type_qualifier
2      ::= VOID
3      |   qualifier
4
5  qualifier
6      ::= (IDENTIFIER (PERIOD IDENTIFIER)* (LBRACKET RBRACKET))
7      |   IDENTIFIER (PERIOD IDENTIFIER)*
```

4.7.2 Context-beperkingen

De qualifiers kennen geen directe context-beperkingen. Er gelden wel context-beperkingen die voort komen uit de wijze waarop de regels gebruikt worden. Deze beperking staan dan bij de desbetreffende regels beschreven.

4.7.3 Semantiek

Een qualifier geldt als een aanduiding van een type. Een qualifier bestaat dus uit nul of meer namespace identifiers, gescheiden door punten, gevolgd door een punt en de identifier van een type. De regel *type_qualifier* introduceert hierbij de aanduiding *VOID* wat naar niets wijst.

4.8 Lexicon

Op het lexicale niveau bestaat het programma uit een set tokens (die gedeeltelijk onder sectie tokens staan), commentaar regels, witte ruimte en literals.

4.8.1 Syntax

Hieronder volgt de syntax beschrijving in *Extended Bachus Naur Form* van alle lexicon gerelateerde regels.

```
1  IDENTIFIER
2      ::= (LETTER | '_') (LETTER | DIGIT | '_')*
3
4  STRING
5      ::= ('"' (QUOTED_CHARACTER | ~('"' | '\\'))* '"')
6
7  CHAR
8      ::= '\'' (QUOTED_CHARACTER | ~('\'' | '\\')) '\''
9
10 QUOTED_CHARACTER
11     ::= '\\' .
12
13 NUMBER
14     ::= (MINUS | PLUS)? DIGIT+
15
16 FLOAT
17     ::= (MINUS | PLUS)? DIGIT+ '.' DIGIT+
18
19 DIGIT
20     ::= ('0'..'9')
21
22 LOWER
23     ::= ('a'..'z')
24
25 UPPER
26     ::= ('A'..'Z')
27
28 LETTER
29     ::= LOWER
30     |   UPPER
31
32 COMMENT
33     ::= '//' .* '\n'
34     |   '/*' .* '*/'
35
36 WHITESPACE
37     ::=   (' ' | '\t' | '\f' | '\r' | '\n')+
```

4.8.2 Context-beperkingen

De lexicon regels kennen geen context beperkingen.

4.8.3 Semantiek

Er zijn enkele semantische regels voor het lexicon. Deze zijn als volgt:

- De waarde van een *STRING* is gelijk aan de achtereenvolgende tekens tussen de beide quotes in. Hierbij dienen de waarden van de *QUOTED_CHARACTER* tekens te worden meegenomen.

- De waarde van een *CHAR* is gelijk aan het teken tussen de beide enkele quotes in. Indien deze een *QUOTED_CHARACTER* bevat is dit gelijk aan de waarde van die specifieke *QUOTED_CHARACTER*.
- De waarde van een *QUOTED_CHARACTER* is als volgt:

  \' De waarde is een enkele quote (').

  \" De waarde is een dubbele quote (").

  **** De waarde is een backslash (\).

  \0 De waarde is een null byte.

  \a De waarde is gelijk aan een systeem alert. Dit is over het algemeen een pieptoon gemaakt door de systeem speaker.

  \b Dit genereert een backspace.

  \f Dit heeft een formfeed als waarde.

  \n De waarde is een newline.

  \r De waarde is een carriage return.

  \t Hiervan is de waarde een tab.

  \v Dit heeft een verticale tab als waarde.

  \x<4 hexadecimale tekens> De waarde hiervan is het *ASCII* teken corresponderend met de hexadecimale reeks.

  \u<4 hexadecimale tekens> De waarde hiervan is het *Unicode* teken corresponderend met de hexadecimale reeks.

  \U<8 hexadecimale tekens> De waarde hiervan is het *Unicode* teken corresponderend met de hexadecimale reeks.

  Alle overige waarden zijn niet toegestaan.

- Een *NUMBER* heeft de numerieke waarde van de bijbehorende tekenreeks. Hierin gelden de mogelijke + en - als respectievelijk de positieve of de negatieve waarde van het getal.
- De waarde van een *FLOAT* is gelijk aan het drijvende kommagetal dat de bijbehorende tekens vormen. Ook hierin gelden de mogelijke + en - als respectievelijk de positieve of de negatieve waarde van het getal.

4.9 Tokens

De taal VNVD gebruikt ook nog een aantal tokens, die gebruikt worden in de geschreven code. Deze zijn als volgt:

```
1  ABSTRACT = 'abstract'
2  ADDEVENT = '+='
3  AND = '&&'
4  AS = 'as'
5  BASE = 'base'
6  BECOMES = '='
7  CAREBOX = 'carebox'
8  CATCH = 'catch'
9  CLASS = 'class'
10 COLON = ':'
```

```
11  COMMA = ','
12  CONST = 'const'
13  DCOLON = '::'
14  DIVISION = '/'
15  ELSE = 'else'
16  ENUM = 'enum'
17  EQ = '=='
18  EXTENDS = 'extends'
19  FALSE = 'false'
20  FI = 'fi'
21  FINALLY = 'finally'
22  FOR = 'for'
23  GE = '>'
24  GEQ = '>='
25  IF = 'if'
26  IMPLEMENTS = 'implements'
27  IN = 'in'
28  INITONLY = 'initonly'
29  INTERFACE = 'interface'
30  INTERNAL = 'internal'
31  IS = 'is'
32  LAND = '&'
33  LBRACKET = '['
34  LCURLYBRACE = '{'
35  LE = '<'
36  LEQ = '<='
37  LOR = '|'
38  LPAREN = '('
39  LXOR = '^'
40  MINUS = '-'
41  MOD = '%'
42  MULTIPLY = '*'
43  NAMESPACE = 'namespace'
44  NEQ = '!='
45  NEW = 'new'
46  NOT = '!'
47  NULL = 'null'
48  OR = '||'
49  OVERRIDE = 'override'
50  PERIOD = '.'
51  PLUS = '+'
52  PRINT = 'print'
53  PRIVATE = 'private'
54  PROTECTED = 'protected'
55  PUBLIC = 'public'
56  QUESTION = '?'
57  RBRACKET = ']'
58  RCURLYBRACE = '}'
59  READ = 'read'
60  READ = 'read'
61  REMEVENT = '-='
62  RETURN = 'return'
63  RPAREN = ')'
64  SEMICOLON = ''
65  STATIC = 'static'
66  STATIC = 'static'
67  THEN = 'then'
68  THIS = 'this'
69  THROW = 'throw'
70  TILDE = '~'
71  TRUE = 'true'
72  TRY = 'try'
```

```
73  USING = 'import'
74  VIRTUAL = 'virtual'
75  VOID = 'void'
76  WHILE = 'while'
77  WRITE = 'write'
```

5 Vertaalregels

In dit hoofdstuk zullen de vertaalregels van VNVD worden besproken. VNVD bestaat uit het object georienteerde gedeelte voor de structuur van een programma en het uitvoerbare gedeelte, de werkelijke opcodes. Voor de vertaalregels zal alleen het uitvoerbare gedeelte in acht worden genomen, omdat er voor de structuur van het programma geen goede vertaalregels zijn op te stellen. We onderscheiden dus alleen de code functies op het niveau van methodes en lager. Verder zijn sommige vertaalregels op een hoger abstractieniveau dan de eigenlijke gegenereerde code. Dit, omdat de regels anders te groot en minder begrijpelijk zouden worden.

Code functies in VNVD:

Methode	run M	Draait de methode P, beginnend met een lege stack en eindigend met ofwel een lege stack of het resultaat van de methode op de stack.
Expressie	evaluate E	Evalueert de expressie E. Omdat VNVD een expressietaal is, valt vrijwel alles onder deze code functie.
Declaratie	elaborate D	Declareer een variabele in de huidige scope.

5.1 Methode

Een methode M bestaat uit een expressie die uitgevoerd wordt. We kunnen dus concluderen dat het draaien van het methode als het volgende gezien kan worden.

```
1  run[E] =
2      evaluate E
3      ret
```

5.2 Expressie

5.2.1 Ternary expressie

```
1  evaluate[if E1 then E2 else E3 fi] =
2      evaluate E1
3      brfalse g
4      evaluate E2
5      br h
6  g:  evaluate E3
7  h:
```

Zoals hierboven beschreven zal een ternary expressie als eerste E1 evalueren. Deze levert een boolean waarde op. Als deze false is wordt een branch gedaan naar E3, anders wordt E2 geëvalueerd. Dan zal na E2 worden gebranched naar het einde van de expressie.

5.2.2 While

```
1  evaluate[while E1 E2] =
2  g:  evaluate E1
3      brfalse h
4      evaluate E2
```

```
5       br g
6   h:
```

De while wordt net zolang uitgevoerd totdat E1 false oplevert. Dat is hierboven ook te zien. E1 wordt eerst geëvalueerd, waarna er naar het einde van de expressie wordt gesprongen als E1 false is. Als E1 true is wordt E2 uitgevoerd en wordt daarna weer gebranched naar E1.

5.2.3 Foreach

```
1   evaluate[for D in E1 E2] =
2       elaborate D
3       evaluate E1
4       callvirt IEnumerator IEnumerable.GetEnumerator()
5   g:  dup
6       callvirt Boolean IEnumerator.MoveNext()
7       brfalse h
8       dup
9       callvirt Object IEnumerator.get_Current()
10      castclass typeof(D)
11      stloc D
12      evaluate E2
13      br g
14  h:  pop
```

Foreach loopt over alle elementen van een IEnumerable object heen. E1 zal een object opleveren dat IEnumerable is. Hier wordt de IEnumerator van verkregen, waarna net zolang MoveNext() hierop zal worden aangeroepen gevolgd door het evalueren van E2, totdat MoveNext() false oplevert. Dan wordt er naar het einde van de loop gesprongen en een laatste pop uitgevoerd om de instantie van IEnumerator van de stack te halen.

5.2.4 Invocatie

```
1   evaluate[E1 :: I ( E2 .. En )] =
2       evaluate[E1]
3       evaluate[E2]
4       ...
5       evaluate[En]
6       call typeof(E1)::I
```

Deze vorm van invocatie wordt gebruikt wanneer de methode die aangeroepen wordt niet virtual is of wanneer expliciet, met het keyword *base*, een methode uit de base class wordt aangeroepen. Eerst wordt E1 geëvalueerd. Deze levert een instantie van een klasse op waarop de methode zal worden aangeroepen. Daarna worden de meegegeven argumenten, E2..En, geëvalueerd. Als dit is gebeurd wordt de methode zelf aangeroepen. Het kiezen welke methode exact aangeroepen moet worden gebeurd op basis van de meegegeven argumenten. Er is overloading mogelijk.

```
1   evaluate[E1 :: I ( E2 .. En )] =
2       evaluate[E1]
3       evaluate[E2]
4       ...
5       evaluate[En]
6       callvirt typeof(E1)::I  I is virtual en geen call naar base
            class
```

Wanneer een methode is gemarkeerd als virtual wordt in plaats van call een callvirt instructie gebruikt. Verder is deze hetzelfde als de eerder beschreven

invocatie.

5.2.5 Object creatie

```
1  evaluate[new I ( E1 .. En )] =
2      evaluate[E1]
3      ...
4      evaluate[En]
5      newobj typeof(I)::.ctor
```

Voor het instantiëren van objecten wordt dit gebruikt. Eerst worden de meegegeven argumenten E1..En geëvalueerd. Daarna wordt de instructie newobj gegeven die de instantie construeert.

5.2.6 Assignment

```
1  evaluate[I = E1] =
2      evaluate E1
3      stloc I
```

In dit geval wordt de waarde van expressie E1 toegekend aan de identifier I. De instructie stloc wordt gebruikt als I wijst naar een lokale variabele.

```
1  evaluate[I = E1] =
2      evaluate E1
3      starg I I is methode argument
```

Idem aan bovengenoemde, behalve dat in dit geval een waarde aan een methode argument wordt toegekend.

```
1  evaluate[E1 :: I = E2] =
2      evaluate E1
3      evaluate E2
4      stfld typeof(E1)::I
```

Wanneer een waarde aan een field wordt toegekend wordt de instructie stfld gebruikt. Eerst wordt E1 geëvalueerd die de waarde van de instantie van een klasse op de stack achterlaat. Daarna wordt E2 geëvalueerd en als laatste wordt stfld aangeroepen.

```
1  evaluate[E1 [ E2 ] = E3] =
2      evaluate E1
3      evaluate E2
4      evaluate E3
5      stelem typeof(E1)
```

In dit geval wordt een element van een array op een bepaalde waarde gezet. Hiertoe moet de eerste waarde op de stack de array zelf zijn, E1, de tweede moet de index zijn van het element in de array, in dit geval E2, en de derde de waarde die toegekend wordt. Daarna wordt de instructie E1 uitgevoerd om de waarde in de array te zetten.

5.2.7 Array creatie

```
1  evaluate[new I [ E1 ]] =
2      evaluate E1
3      newarr typeof(I)
```

Het maken van een array gebeurt door eerst de grootte van de array op de stack te zetten door E1 te evalueren. E1 laat een integer achter. Vervolgens wordt newarr uitgevoerd.

5.2.8 Array index

```
1  evaluate[E1 [ E2 ]] =
2      evaluate E1
3      evaluate E2
4      ldelem typeof(E1)
```

Om een element uit een array te verkrijgen moet eerst de array op de stack staan, daarna de index uit de array en vervolgens moet ldelem worden uitgevoerd. In dit geval zorgt E1 voor de array en E2 voor de integer van de index.

5.2.9 Overloadable binary operators

```
1  evaluate[E1 O E2] =
2      evaluate E1
3      evaluate E2
4      O
```

In dit geval stelt O de operator voor die toegepast wordt. E1 en E2 zijn van hetzelfde type en bovendien een primitive type. De operator kan bijvoorbeeld vermenigvuldigen zijn, *mult*, of logic or, *or*.

```
1  evaluate[E1 O E2] =
2      evaluate E1
3      evaluate E2
4      call <T> typeof(E1)::op_O(typeof(E1), typeof(E2))
```

Als de types van E1 en E2 niet primitive zijn, maar zij wel operator overloading hebben geïmplementeerd wordt dit toegepast. De methode wordt aangeroepen van de betreffende operator.

5.2.10 Overloadable unary operators

```
1  evaluate[O E1] =
2      evaluate E1
3      O
```

```
1  evaluate[O E1] =
2      evaluate E1
3      call <T> typeof(E1)::op_O(typeof(E1))
```

Hier geldt bijna hetzelfde als voor de binary operators. Het verschil is dat nu maar één expressie wordt geëvalueerd waarna de operator wordt aangeroepen. Ook deze kan overloaded zijn.

5.2.11 Conditional and

```
1  evaluate[E1 && E2] =
2      evaluate E1
3      brfalse g
4      evaluate E2
5      br h
6  g:  ldc.i.0
7  h:
```

Deze operator kan niet overloaded worden, en wordt bovendien lazy geëvalueerd. Eerst wordt E1 berekend, als deze false is kan de waarde van de hele expressie nooit meer true worden, dus wordt gelijk naar het einde gesprongen. Anders wordt E2 geëvalueerd en wordt naar het einde gesprongen.

5.2.12 Conditional or

```
1  evaluate[E1 || E2] =
2      evaluate E1
```

```
3       brtrue g
4       evaluate E2
5       br h
6   g:  ldc.i.1
7   h:
```

Hier geldt hetzelfde als voor de conditional and, met het verschil dat wanneer E1 true oplevert de expressie E1 —— E2 altijd true zal zijn, en dus E2 niet meer geëvalueerd hoeft te worden.

5.2.13 Variabele gebruikt

```
1  evaluate[I] =
2      ldarg.I
```

Deze code template is voor het laden van een methode argument. Hiervoor wordt ldarg gebruikt.

```
1  evaluate[I] =
2      ldloc.I
```

Voor het laden van een lokale variabele wordt de instructie ldloc gebruikt.

5.2.14 Literals

```
1  evaluate[L] =
2      ldstr L
```

```
1  evaluate[L] =
2      ldc.i4 L
```

Hier staan twee voorbeelden gegeven voor het gebruik van literals. In de bovenste code template wordt een literal string geladen met de instructie *ldstr*. In die daaronder wordt een integer geladen met *ldc.i4*. Andere instructies om literals te laden zijn: *ldc.i8* voor een long, *ldc.r4* voor een float, *ldc.r8* voor een double en *ldc.i4* voor een char.

5.2.15 Casting

```
1  evaluate[ ( I ) E1 ] =
2      evaluate E1
3      castclass I
```

Een directe cast gebruikt de instructie *castclass*. Het effect hiervan is dat het type van E1 veranderd wordt naar het type van I. Wanneer het casten niet mogelijk is wordt een exceptie gegooid.

```
1  evaluate[E1 as I] =
2      evaluate E1
3      isinst I
```

Het verschil met bovenstaande cast is dat deze geen exceptie gooit wanneer het casten niet mogelijk is. Hiervoor wordt dan ook een andere instructie gebruikt, namelijk *isinst*.

```
1  evaluate[E1 is I] =
2      evaluate E1
3      isinst I
4      ldnull
5      cgt_un
```

Deze expressie bekijkt of een E1 van een bepaald type is. Hiertoe wordt eerst *isinst* uitgevoerd, waarna de uitkomst hiervan wordt vergeleken met null.

5.2.16 Read

```
1  evaluate[read(I)] =
2      call ConsoleKeyInfo Console::ReadKey()
3      call Char ConsoleKeyInfo::get_KeyChar()
4      stloc I
5      ldloc I
```

Bovenstaande read expressie wordt gebruikt voor het inlezen van een character van de standaardinvoer. Omdat er maar één wordt ingelezen, moet de expressie de waarde weer achterlaten op de stack. Hier zorgt de laatste *ldloc* voor.

Het inlezen van een string of een integer gebeurt op ongeveer dezelfde manier. Het verschil is dat voor een string de methode *Console::ReadLine()* wordt aangeroepen, en voor een integer de methode *Console::ReadLine()* gevolgd door *Int32::Parse(String)*.

```
1  evaluate[read(I1 .. In)] =
2      call ConsoleKeyInfo Console::ReadKey()
3      call Char ConsoleKeyInfo::get_KeyChar()
4      stloc I1
5      ...
6      call ConsoleKeyInfo Console::ReadKey()
7      call Char ConsoleKeyInfo::get_KeyChar()
8      stloc In
```

Bij het inlezen van meerdere variabelen worden deze opgesplitst in meerdere aanroepen. De *ldloc* wordt weggelaten omdat het type van deze expressie *void* is. Ook hier zijn weer verschillende versies voor voor het inlezen van meerdere strings en integers.

5.2.17 Write

```
1  evaluate[write(E1)] =
2      evaluate E1
3      dup
4      call Void Console::Write(Object)
```

Bij het schrijven naar de standaarduitvoer zal eerst E1 worden geëvalueerd. Daarna zal de instructie *dup* worden uitgevoerd, omdat de expressie een waarde moet achterlaten op de stack. Als laatste wordt *Console::Write(Object)* aangeroepen die de expressie naar de standaarduitvoer schrijft.

```
1  evaluate[write(E1 .. En)] =
2      evaluate E1
3      call Void Console::Write(Object)
4      ...
5      evaluate En
6      call Void Console::Write(Object)
```

Bij het schrijven van meerdere expressies naar de standaard uitvoer wordt de instructie *dup* weggelaten. Verder is deze identiek aan bovenstaande, behalve dat *Console::Write(Object)* voor elke expressie wordt aangeroepen.

5.3 Declaratie

5.3.1 Lokale variabele

Het declareren van een lokale variabele levert niet gelijk instructies op die uitgevoerd worden. Het effect van de declaratie is dat het type gebind wordt aan de

identifier en dat in de sectie locals van de methode een variabele gedeclareerd wordt.

5.3.2 Lokale constante

Het enige effect van een lokale constante is dat de compiler de waarde ervan gelijk opslaat in de checkerfase. In de generator fase wordt bij elke keer dat deze gebruikt wordt direct de waarde weggeschreven in plaats van dat er geheugen wordt vrijgemaakt om de waarde in op te slaan.

6 Beschrijving van de programmatuur

Naast de door ANTLR gecreëerde klassen bevat de code van de VNVD compiler nog enkele andere stukken code. Deze klassen staan hier per bestand en per klasse beschreven om zo een goed beeld te geven van de programmatuur.

Op de meegeleverde cd-rom is ook een volledige documentatie gegenereerd uit de XML code documentation comments te vinden. In deze documentatie wordt ook uitgebreid ingegaan op de functionaliteit van de diverse members van de klassen.

6.1 AbstractHelper.cs

Hierin staat de klasse *Vnvd.AbstractHelper* gedefinieerd. Deze klasse vormt de basis voor de zogenaamde TreeWalker Helpers die de VNVD compiler gebruikt. In deze klasse worden alle functies gedefinieerd die vanuit de TreeWalker kunnen worden aangeroepen, daarnaast bevat zij een aantal algemene constructies voor de TreeWalkers.

6.2 CheckerHelper.cs

Dit bestand bevat de definities van twee klassen, namelijk *Vnvd.CheckerHelper* en *Vnvd.CheckerException*. De eerst genoemde is de exceptie die de checker gebruikt als zij inconsistenties of andere problemen aantreft. De tweede klasse is de uitbreiding van de *Vnvd.AbstractHelper* klasse voor de checker. Deze klasse voert het eigenlijke werk van de checker uit.

6.3 CheckerManual.cs

De definitie van de klasse *Vnvd.CheckerManual* geschiedt in dit bestand. Deze klasse wordt voor de TreeWalker van de checker gedraaid. In deze eerste pass van de checker component worden namespaces, klassen en hun members gedefiniëerd, zodat zij probleemloos gebruikt kunnen worden.

6.4 EntryType.cs

In het bestand *EntryType.cs* wordt de enumeratie *Vnvd.EntryType* gedeclareerd. Deze enumeratie bevat de typen die een *Vnvd.TreeNode* kan hebben.

6.5 GeneratorHelper.cs

Dit bestand bevat de klasse *Vnvd.CheckerHelper*. Deze klasse is de uitbreiding van de *Vnvd.AbstractHelper* klasse voor de code generator. Hierin wordt het eigenlijke werk van de code generator uitgevoerd.

6.6 GeneratorManual.cs

Hierin staat de definitie van de klasse *Vnvd.GeneratorManual*. Deze klasse wordt voor de TreeWalker van de code generator gedraaid. In deze eerste pass van de code generator component wordt de code voor namespaces, klassen en hun members gegenereerd, zodat zij probleemloos gebruikt kunnen worden.

6.7 IdEntry.cs

De definitie van de klasse *Vnvd.IdEntry* bevindt zich in dit bestand. Deze klasse is een representatie voor een item in een *Vnvd.ISymTab*. Deze klasse bevat onder andere een link naar de node in de *abstract syntax tree* waarin dit item gedeclareerd wordt.

6.8 ISymTab.cs

Dit bestand bevat de definitie van de klasse *Vnvd.ISymTab*. Dit is de interface gebruikt voor een symbol table. De basis functionaliteiten voor een symbol table staan hierin gedefinieerd.

6.9 LibraryChecker.cs

De hierin gedefinieerde klasse *Vnvd.LibraryChecker* is bedoeld om te controleren of externe objecten bestaan. Met behulp van deze klasse kunnen aanroepen naar objecten, methoden en dergelijken die elders in het .NET framework staan gecontroleerd worden. De checker gebruikt deze klasse om ervoor te zorgen dat alle definities bestaan. De code generator gebruikt vervolgens de resultaten om in de assembly de juiste links te plaatsen.

6.10 Pair.cs

De klasse *Vnvd.Pair* is een representatie voor een tuple van twee variabelen. Deze klasse wordt vooral gebruikt om interne presentatie te vergemakkelijken.

6.11 Parameter.cs

De klasse *Vnvd.Paramter* is een representatie voor een tuple van een volledige qualifier, geïmplementeerd als een *Vnvd.Qualifier*, en een identifier. De eerste is in dit geval een type en de tweede de naam van de parameter. Deze klasse wordt vooral gebruikt om interne presentatie te vergemakkelijken.

6.12 Qualifier.cs

De klasse *Vnvd.Qualifier* is een representatie voor een volledige qualifier. Dat is een referentie naar een namespace, klasse of methode. Deze klasse wordt vooral gebruikt om interne presentatie te vergemakkelijken.

6.13 StringHelper.cs

Dit bestand bevat de statische klasse *Vnvd.StringHelper*. In deze klasse staan enkele methoden voor het gebruik van string literals in de te compileren code. Zo staan de methoden die escape sequences kunnen omzetten naar een echte string in deze klasse.

6.14 SymbolTable.cs

In dit bestand staat de klasse *Vnvd.SymbolTable* en de bijbehorende klasse *Vnvd.SymbolTableException*. Deze klasse bevat de in VNVD gebruikte implementatie van een symbol table en is dan ook een uitbreiding van *Vnvd.ISymTab*.

In VNVD kent elke methode zijn eigen symbol table, waarin de variabelen die in de scope van die methode gedeclareerd worden zijn opgenomen. Naar members van de klasse zelf of een van zijn superklassen wordt gerefereerd via de *this* of de *base* variabele. Deze hoeven dus niet in de symbol table meegenomen te worden.

6.15 TreeNode.cs

De klasse *Vnvd.TreeNode* staat in dit bestand. Dit is de in de TreeWalkers gebruikte versie van de *abstract syntax tree* node. Naast de standaard functionaliteit bevat deze ook properties voor de entry in de symbol table, het type van de entry, het (return) type van de node en de waarde van de node.

Aan een TreeNode van, bijvoorbeeld, een literal hangt dus het type van de literal en zijn waarde. Bij de TreeNode van een variabele zal, naast zijn type, de entry in de symbol table van deze variabele te vinden zijn.

6.16 TreeNodeAdapter.cs

Dit bestand bevat de klasse *Vnvd.TreeNodeAdaptor*. Naast dat deze klasse om historische redenen een spelfout in de naam heeft is dit ook de klasse die ANTLR gebruikt om *Vnvd.TreeNode* objecten aan te kunnen maken.

6.17 Vnvd.cs

De zogenaamde *main()* functie van VNVD is in dit bestand gedefinieerd. Deze maakt deel uit van de klasse *Vnvd.Vnvd*. Dit is de klasse die de compiler start en aanstuurt.

6.18 Antlr/VnvdChecker.cs

Dit bestand bevat de door ANTLR gegenereerde TreeWalker voor de checker. De ANTLR bron hiervan is te vinden in de apendices.

6.19 Antlr/VnvdGenerator.cs

Dit bestand bevat de door ANTLR gegenereerde TreeWalker voor de code generator. De ANTLR bron hiervan is te vinden in de apendices.

6.20 Antlr/VnvdLexer.cs

Dit bestand bevat de door ANTLR gegenereerde lexer. De ANTLR bron hiervan is te vinden in de apendices.

6.21 Antlr/VnvdParser.cs

Dit bestand bevat de door ANTLR gegenereerde parser. De ANTLR bron hiervan is te vinden in de apendices.

6.22 Properties/AssemblyInfo.cs

In dit bestand staan algemene definities die nodig zijn voor een C# applicatie. Deze specificaties zijn niet VNVD specifiek.

6.23 UserDefined/CustomBinder.cs

De in dit bestand gedefinieerde klasse *Vnvd.UserDefined.CustomBinder* zorgt voor de zogenaamde binding van typen. Deze klasse is een uitbreiding op de standaard functionaliteit van C# waarin niet alleen de functionaliteiten uit externe assemblies worden gevonden, maar ook die in de bestanden die nu gecompileerd worden.

6.24 UserDefined/ErrorType.cs

De klasse in dit bestand, *Vnvd.UserDefined.ErrorType*, is het type dat wordt gegeven aan nodes in de *abstract syntax tree* waarbij een fout is ontstaan. De checker gebruikt dit type om de rest van de code wel door de checker te kunnen halen, ondanks een gedetecteerde fout. Zo kunnen er in een keer meer en betere foutmeldingen worden gegeven, voordat de checker moet onderbreken. Deze klasse breid *Vnvd.UserDefined.UserType* uit.

6.25 UserDefined/IMethodOrConstructor.cs

De interface die gebruikt wordt voor de klassen *Vnvd.UserDefined.UserMethod* en *Vnvd.UserDefined.UserConstructor* wordt in dit bestand gedefinieerd. Zo kunnen generieke methoden, die voor zowel methoden als constructoren gelden, op een wijze worden afgehandeld.

6.26 UserDefined/MethodType.cs

In dit bestand wordt de klasse *Vnvd.UserDefined.MethodType* gedefinieerd. Deze klasse representeert een delegate naar een methode. Dat wil zeggen, een pointer naar een methode. In de *abstract syntax tree* kan deze klasse gebruikt worden om het type van een node een methode invocatie te maken.

6.27 UserDefined/NullType.cs

Dit bestand bevat de klasse *Vnvd.UserDefined.NullType* die het een object met waarde null representeert. Deze klasse maakt een uitbreiding van de klasse *Vnvd.UserDefined.UserType*.

6.28 UserDefined/UserConstructor.cs

De hierin gedefinieerde klasse *Vnvd.UserDefined.UserConstructor* representeert een constructor type van een klasse in de te compileren code. Deze klasse maakt gebruik van de interface *Vnvd.UserDefined.IMethodOrConstructor*.

6.29 UserDefined/UserEnum.cs

Dit bestand bevat de definitie van de klasse *Vnvd.UserDefined.UserEnum*. Deze klasse is een representatie van een enumeratie type in de te compileren code. De klasse *Vnvd.UserDefined.UserType* wordt door deze klasse uitgebreid.

6.30 UserDefined/UserField.cs

De in dit bestand gedefinieerde klasse *Vnvd.UserDefined.UserField* wordt gebruikt om een in de te compileren code gedefinieerd veld te representeren.

6.31 UserDefined/UserLocal.cs

De in dit bestand gedefinieerde klasse *Vnvd.UserDefined.UserLocal* wordt gebruikt om een in de te compileren code gedefinieerde lokale variabele te representeren.

6.32 UserDefined/UserMethod.cs

De in dit bestand gedefinieerde klasse *Vnvd.UserDefined.UserMethod* wordt gebruikt om een in de te compileren code gedefinieerde methode te representeren.

6.33 UserDefined/UserParameter.cs

De in dit bestand gedefinieerde klasse *Vnvd.UserDefined.UserParameter* wordt gebruikt om een in de te compileren code gedefinieerde parameter te representeren.

6.34 UserDefined/UserType.cs

De klasse *Vnvd.UserDefined.UserType*, die gedefinieerd is in dit bestand, is een representatie van een in de te compileren code gedefinieerd type. Deze user-defined typen zijn (abstracte) klassen en interfaces. Daarnaast gebruiken de typen *Vnvd.UserDefined.UserEnum*, *Vnvd.UserDefined.ErrorType* en *Vnvd.UserDefined.NullType* deze klasse als superklasse.

7 Testplan en testresultaten

Om te zorgen dat de VNVD compiler niet alleen volledig functioneel is, maar ook correct functioneert, zijn er een aantal test programma's gedefinieerd. Een aantal van deze tests zijn zeer uitgebreid en een aantal zijn kleine testjes. Deze tests zijn ook allemaal op soft-copy meegeleverd.

Van elke test zijn de opzet van de test en de resultaten van de test besproken. Hierdoor zijn de tests goed reproduceerbaar en de resultaten makkelijk toetsbaar.

7.1 BoterKaasEieren

De testapplicatie *Boter, Kaas en Eieren* is tevens de uitgebreide testapplicatie, zoals in de appendices vermeld. Dit is een applicatie om een ouderwets spelletje boter, kaas en eieren te spelen tegen een andere persoon, een domme computer of een intelligente *AI*. Het is overigens ook mogelijk om twee *AI*s tegen elkander te laten spelen.

7.1.1 Testopzet

De broncode bestaat uit een aantal bestanden, die naast op de meegeleverde CD-Rom ook in de bijbehorende appendix te vinden zijn. Deze kunnen gecompileerd worden met de volgende opdracht:

```
1 Vnvd.exe --ref:System --ref:System.Windows.Forms --ref:System.
    Drawing --out:BKE.exe Bord.vnvd Mark.vnvd MensSpeler.vnvd Spel.
    vnvd ComputerSpeler.vnvd IStrategie.vnvd Speler.vnvd BkeGui.
    vnvd IMessageListener.vnvd BesteStrategie.vnvd DommeStrategie.
    vnvd
```

De applicatie kan daarna uitgevoerd worden door de executable *BKE.exe* op te starten.

7.1.2 Testresultaten

Deze applicatie valt succesvol te compileren. De uitvoer van de VNVD compiler is dan ook niet erg indrukwekkend voor deze applicatie:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
```

Daarnaast valt de executable te controleren met de assembly verifier die bij de virtuele machine hoort, ook dit gaat naar behoren:

```
1 Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2 Copyright (C) Microsoft Corporation. All rights reserved.
3
4 All Classes and Methods in BKE.exe Verified.
```

Tijdens het draaien van de applicatie werkt het geheel ook naar behoren. Bij wijze van steekproef vallen er een aantal spelletjes te spelen op verscheidene configuraties.

7.2 Bounce

De test *Bounce* is niet alleen een taal test maar ook een test van de interactie met de onderliggende libraries.

7.2.1 Testopzet

De *Bounce* test zal eerst een library compileren, genaamd *DebugLib.dll*. Daarna zal er een executable *Bounce.exe* gecompileerd worden. Deze compilatie opdracht is als volgt:

```
1 Vnvd.exe DebugLib.vnvd --out:DebugLib.dll
2 Vnvd.exe Bounce.vnvd --out:Bounce.exe --ref:System.Windows.Forms --
      ref:System --ref:System.Drawing --ref:DebugLib.dll
```

Daarna kan de applicatie gestart worden door *Bounce.exe* te executeren. Bij executie zal de applicatie een rode bal tekenen die tegen de randen van het venster stuitert. Daarnaast zullen er regels op de standard output worden geprint.

Het correct werken van deze test betekent dat VNVD in staat is geweest globale libraries aan te koppelen, die gebruikt worden voor het tekenen en het venster, en ook een lokaal library, die gebruikt wordt om tekst op de standard output af te drukken.

7.2.2 Testresultaten

De volledige compilatie van het *Bounce* programma geeft een tweetal tevreden compilatie uitvoeren:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
3
4 Vnvd Compiler v1.3.3.8
5 ========================
```

Daarnaast kan ook de assembly verifier geen problemen aanwijzen. De compilatie is dus in ieder geval geslaagd. Als de gecompileerde applicatie opgestart wordt begint de animatie van de heen en weer stuiterende bal te draaien, terwijl er op de console bij elke collision een melding wordt gezet. De applicatie functioneert dus naar wens.

7.3 ChatClient

De *ChatClient* applicatie is een client voor de server van de *ChatServer* applicatie. Deze applicatie levert een textitGUI voor de desbetreffende chat server. Hiermee kunnen berichtjes verstuurt worden naar andere clients.

7.3.1 Testopzet

Om de code van deze test applicatie te compileren naar de executable *Client.exe* kan de volgende opdracht uitgevoerd worden:

```
1 Vnvd.exe --out:Client.exe --ref:System --ref:System.Drawing --ref:
      System.Windows.Forms Client.vnvd Gui.vnvd IMessageListener.vnvd
```

Bij executie zal de applicatie een interface openen. In deze interface kan eerst een naam opgegeven worden. Vervolgens zullen alle berichten naar alle andere clients doorgestuurd worden, met de naam als prefix. Voor de werking van de applicatie is het wel vereist dat de server draait.

7.3.2 Testresultaten

Ook deze applicatie valt succesvol te compileren. Dit geeft de standaard compiler uitvoer voor een correct werkende applicatie:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
```

Ook de *.NET Framework PE Verifier* is tevreden over het product van de VNVD compiler:

```
1 Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2 Copyright (C) Microsoft Corporation. All rights reserved.
3
4 All Classes and Methods in Client.exe Verified.
```

Daarnaast valt in een aantal praktijk tests berichten over de server te sturen. Deze test blijkt dus geslaagd.

7.4 ChatServer

Deze test applicatie is de serverende wederhelft van de *ChatClient*.

7.4.1 Testopzet

Deze applicatie kan met het volgende commando door de VNVD compiler gehaald worden:

```
1 Vnvd.exe --out:Server.exe --ref:System ClientHandler.vnvd Server.
      vnvd
```

Als de applicatie *Server.exe* vervolgens geëxecuteerd wordt zal er een server opgestart worden. Door het opstarten van de client applicatie wordt er verbinding gemaakt met de gestartte server.

7.4.2 Testresultaten

De resultaten naar de test van de werking van deze applicatie zijn te vinden onder de test van de client. Om echter te tonen dat de compilatie ook succes vol liep, hierbij de compiler uitvoer:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
```

Ook de *.NET Framework PE Verifier* is tevreden over de gecompileerde server:

```
1 Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2 Copyright (C) Microsoft Corporation. All rights reserved.
3
4 All Classes and Methods in Server.exe Verified.
```

7.5 Tests/ContextIncorrect.vnvd

Deze test zal de context checker aan de tand voelen. Dat betekent dat er verkeerde typen gebruikt zullen worden om zo te kijken of de checker dit opmerkt.

7.5.1 Testopzet

Deze test bestaat uit een bestand, genaamd *ContextIncorrect.vnvd*. Dit bestand bevat een aantal context gerelateerde fouten. Om dit bestand te compileren kan de VNVD compiler als volgt worden aangeroepen:

```
1 Vnvd.exe ContextIncorrect.vnvd
```

Deze testapplicatie is een VNVD implementatie van testprogramma *A.3* in de practicumhandleiding.

7.5.2 Testresultaten

Door de vele context fouten geeft de VNVD compiler voor deze test een zeer grote hoeveelheid uitvoer. Aan de hand hiervan zijn gelukkig wel de gemaakte fouten aan te wijzen:

```
1  Vnvd Compiler v1.3.3.8
2  ======================
3
4  ContextIncorrect.vnvd   11:11   Left part of assignment is not
       assignable:
5  ContextIncorrect.vnvd   21:16   Types do not match: System.Int32
       and System.Char
6  ContextIncorrect.vnvd   21:20   Types do not match: ~error~.~error~
       and System.Int32
7  ContextIncorrect.vnvd   21:11   Types do not match: System.Boolean
       and ~error~.~error~
8  ContextIncorrect.vnvd   23:11   Types do not match: System.Boolean
       and System.Int32
9  ContextIncorrect.vnvd   23:17   Types do not match: ~error~.~error~
       and System.Char
10 ContextIncorrect.vnvd   25:13   Types do not match: System.Char and
       System.Boolean
11 ContextIncorrect.vnvd   26:11   Types do not match: System.Boolean
       and System.Int32
12 ContextIncorrect.vnvd   27:15   Types do not match: System.Int32
       and System.Char
13 ContextIncorrect.vnvd   29:11   Types do not match: System.Char and
       System.Int32
14 ContextIncorrect.vnvd   30:11   Types do not match: System.Boolean
       and System.Int32
15 ContextIncorrect.vnvd   31:9    Type p cannot be found
16 ContextIncorrect.vnvd   31:12   Type q cannot be found
17 ContextIncorrect.vnvd   34:22   A compound expression must not end
       with a declaration.
18 ContextIncorrect.vnvd   36:20   Types do not match: System.Void and
       System.Int32
19 ContextIncorrect.vnvd   38:13   Types do not match: System.Char and
       System.Void
20 ContextIncorrect.vnvd   39:12   Types do not match: System.Int32
       and System.Void
```

7.6 Tests/CorrectArrays.vnvd

Om te controleren of de implmentatie van arrays naar behoren werkt is deze test nodig. In deze test worden beide wijzen van array definitie getest en er wat mee gewerkt.

7.6.1 Testopzet

Het gemaakte testprogramma kan naar een executable met de naam *CorrectArrays.exe* gecompileerd worden door het volgende commando uit te voeren:

```
1 Vnvd.exe CorrectArrays.vnvd
```

7.6.2 Testresultaten

Wederom heeft de compiler geen fouten te melden. Daarnaast is ook de assembly verifier tevreden over de executable. Door vervolgens de applicatie uit te voeren kan ook de werking getest worden. Deze uitvoer is als volgt:

```
1 2
2 10
3 10
```

Hoewel dit misschien niet zeer veelzeggend is, is dit wel correct. Het eerste getal is namelijk de lengte van een van de arrays. Het tweede getal de optelling van een array met de waarden 5 en 5. Het laatste getal is ook een optelling, maar dan van een array met de getallen 1 tot en met 4.

Ook de test van de arrays verloopt dus naar wens.

7.7 Tests/CorrectConstants.vnvd

De test *CorrectConstants.vnvd* test de constante mogelijkheden aan boord van VNVD. Constanten worden direct vervangen in de assembly.

7.7.1 Testopzet

De executable *CorrectConstants.exe* valt te generen door de VNVD compiler de test te laten compileren. Dit kan met het volgende commando:

```
1 vnvd.exe CorrectConstants.vnvd
```

De test kan vervolgens ook gedraaid worden met behulp van de executable. De test zal wat constanten naar het scherm printen.

7.7.2 Testresultaten

Zoals hieronder te zien is de VNVD compiler tevreden met het resultaat:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
```

Ook de *.NET Framework PE Verifier* geeft aan dat de gecompileerde executable goed is:

```
1 Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2 Copyright (C) Microsoft Corporation. All rights reserved.
3
4 All Classes and Methods in CorrectConstants.exe Verified.
```

Bij het draaien geeft de executable de waarden van de constanten correct weer:

```
1 10
2 Deze test werkt!
3 Inderdaad...
4 1337
```

7.8 Tests/CorrectTest.vnvd

Deze test controleert de functionaliteiten van een expressie taal. Alle facetten van de *Basic Expression Language* komen langs in deze test.

7.8.1 Testopzet

Deze test valt te compileren door de volgende regel uit te voeren:

```
1 Vnvd.exe CorrectTest.vnvd
```

Vervolgens valt de executable *CorrectTest.exe* aan te roepen om het product uit te voeren. De executable zal eerst drie maal een geheel getal verwachten. Daarna dient er ofwel "True", ofwel "False", ingevoerd te worden. Tot slot mag er nog een willekeurig karakter ingevoerd worden. Deze applicatie correspondeert met testprogramma *A.1* in de practicumhandleiding.

7.8.2 Testresultaten

Dit testprogramma valt met succes te compileren en eveneens succesvol te verifiëren. Als de applicatie vervolgens gedraait wordt met de volgende invoer:

```
1 0
2 1
3 1
4 False
5 c
```

Wordt de volgende uitvoer, die ook geleverd wordt, terug gegeven:

```
1 01FalseTrue1FalseTrueaTrue3Trueb
```

De rede dat dit een dergelijke one-liner wordt is dat er geen spaties of newlines uitgevoerd worden. Bij draaiïng vanaf de console kan de uitvoer anders lijken, dit komt dan waarschijnlijk doordat de invoer tussen de uitvoer door staat.

7.9 Tests/CorrectWhileIfFor.vnvd

In deze test worden alle controle structuren uitvoerig getest.

7.9.1 Testopzet

Ook deze test bestaat uit een bestand, *CorrectWhileIfFor.vnvd*, dat als volgt naar de executable *CorrectWhileIfFor.exe* te compileren valt:

```
1 Vnvd.exe CorrectWhileIfFor.vnvd
```

Bij het uitvoeren zal deze executable een aantal waarden naar het scherm uitvoeren. Deze dienen als volgt te zijn:

```
1 Waardering: Onvoldoende!
2 Waardering: Voldoende!
3 Waardering: Goed!
4 Waardering: Slecht!
5 Waardering: Onvoldoende!
6 Waardering: Voldoende!
7 Waardering: Goed!
8 Waardering: Slecht!
```

```
9  14328
10 F
11 v
12 N
13 S
14 V
15 15
```

7.9.2 Testresultaten

De VNVD compiler geeft geen foutmeldingen in zijn uitvoer, wat een goed teken is:

```
1  Vnvd Compiler v1.3.3.8
2  ======================
```

Na een succesvolle compilatie geeft een consultatie van de assembly verifier ook geen slechte tekens:

```
1  Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2  Copyright (C) Microsoft Corporation. All rights reserved.
3
4  All Classes and Methods in CorrectWhileIfFor.exe Verified.
```

Na executie geeft het gecompileerde bestand ook de correcte uitvoer, zoals in de voorgaande sectie gespecificeerd. Deze test is dus correct te bevinden.

7.10 Tests/IncorrectFor.vnvd

Deze test is een van de incorrecte tests. Oftewel, een test die zou moeten falen. In deze specifieke variant wordt getest of de checker wel streng genoeg is bij de for loop.

7.10.1 Testopzet

In de test, *IncorrectFor.vnvd*, wordt geprobeerd een for loop te maken over een niet enummereerbaar object. Oftewel, een object dat geen verzameling is. Om te proberen deze test te compileren roept men VNVD als volgt aan:

```
1  Vnvd.exe IncorrectFor.vnvd
```

7.10.2 Testresultaten

Deze applicatie wil niet compileren, wat natuurlijk het doel is van de test. Ook de gegeven foutmelding is bevredigend:

```
1  Vnvd Compiler v1.3.3.8
2  ======================
3
4  IncorrectFor.vnvd       9:9     Expression used as in-clause is not
       enumerable: System.Object
```

7.11 Tests/IncorrectIf.vnvd

In deze test wordt gepoogd een aantal context-beperkingen van een ternary expressie te doorbreken. Het doel van de test is te zorgen dat de checker de fouten ontdekt en rapporteert.

7.11.1 Testopzet

Het bestand van de test, *IncorrectIf.vnvd*, kan gecompileerd worden. De VNVD compiler geeft vervolgens foutmeldingen terug met de oorzaken van de problemen. De compilatie gaat als volgt:

```
1 Vnvd.exe IncorrectIf.vnvd
```

7.11.2 Testresultaten

Zoals gewenst sputtert de VNVD compiler tegen en geeft de volgende uitvoer:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
3
4 IncorrectIf.vnvd          14:14    Types do not match: System.Int32
      and System.Void
5 IncorrectIf.vnvd          11:9     System.Boolean expected here, got:
      System.Int32
```

7.12 Tests/IncorrectOOP.vnvd

Dit is de eerste van de twee structuur tests. Hierin wordt gekeken of alle object georiënteerde concepten wel naar behoren gecontroleerd worden.

7.12.1 Testopzet

Deze test dient compilatie fouten te geven als men het volgende commando uitvoert:

```
1 Vnvd.exe IncorrectOOP.vnvd
```

7.12.2 Testresultaten

Zoals verwacht is de VNVD compiler niet tevreden met de code en vindt de foute implementaties:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
3
4 11:1     Circular extendation detected at Tests.C and Tests.B
5 17:1     Type: NonExistant not found
6 23:1     Cannot extend interface or value type Tests.Inter
```

7.13 Tests/IncorrectOOP2.vnvd

Dit is de tweede van de twee structuur tests. Hierin wordt gekeken of alle object georiënteerde concepten wel naar behoren gecontroleerd worden.

7.13.1 Testopzet

Deze test dient compilatie fouten te geven als men het volgende commando uitvoert:

```
1 Vnvd.exe IncorrectOOP2.vnvd
```

7.13.2 Testresultaten

Wederom weet de VNVD compiler de vinger op de zere plek te leggen:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
3
4 19:6    Cannot declare abstract method AbstractMethod in a non-
      abstract class: A
5 25:4    No suitable method found to override: Test2
6 31:4    No suitable method found to override: Test2
```

7.14 Tests/IncorrectWhile.vnvd

De test *IncorrectWhile.vnvd* probeert context-beperkingen voor while loops te negeren. De VNVD compiler dient dit af te keuren.

7.14.1 Testopzet

Deze test kan gedraaid worden door VNVD als volgt aan te roepen:

```
1 Vnvd.exe IncorrectWhile.vnvd
```

7.14.2 Testresultaten

Zoals behoort keurt de compiler de code af. Dit gebeurt met de volgende uitvoer:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
3
4 IncorrectWhile.vnvd    11:9    System.Boolean expected here, got:
      System.Int32
5 IncorrectWhile.vnvd    16:11   Types do not match: System.Int32
      and System.Void
```

7.15 Tests/MethodCall.vnvd

Deze test roept enkele methoden aan op zowel objecten als primitieve typen.

7.15.1 Testopzet

Deze test kan simpel weg gedraaid worden door het bestand *MethodCall.vnvd* te compileren en vervolgens de gemaakte executable te executeren. De uitvoer dient dan als volgt te zijn:

```
1 5
2 3
3 15
4 System.Int32[]
5 Hi.
```

Het compileren van *MethodCall.vnvd* naar *MethodCall.exe* kan met de volgende opdracht:

```
1 Vnvd.exe MethodCall.vnvd
```

7.15.2 Testresultaten

Zoals verwacht hebben de VNVD compiler en de *.NET Framework PE Verifier* niets te klagen over de testapplicatie. Daarnaast komt het testprogramma bij een test run met de correcte uitvoer aanzetten, waardoor geconcludeerd kan worden dat de getestte delen naar behoren werken.

7.16 Tests/RuntimeError.vnvd

In deze applicatie wordt een simpele runtime error gemaakt. Deze kan niet afgevangen worden door de VNVD compiler.

7.16.1 Testopzet

Deze applicatie kan gecompileerd worden naar *RuntimeError.exe*. Bij executie van dit programma zal er vervolgens een runtime error optreden. Om de compilatie uit te voeren geeft men het volgende commando op:

```
1 Vnvd.exe RuntimeError.vnvd
```

Deze testapplicatie is een VNVD implementatie van testprogramma *A.4* in de practicumhandleiding.

7.16.2 Testresultaten

Deze applicatie valt succesvol te compileren. De uitvoer van de VNVD compiler is dan ook leeg:

```
1 Vnvd Compiler v1.3.3.8
2 ========================
```

Daarnaast valt de executable te controleren met de assembly verifier die bij de virtuele machine hoort, ook dit gaat naar behoren:

```
1 Microsoft (R) .NET Framework PE Verifier. Version 3.5.21022.8
2 Copyright (C) Microsoft Corporation. All rights reserved.
3
4 All Classes and Methods in RuntimeError.exe Verified.
```

Bij executie gaat het echter fout. De runtime error treed dan daadwerkelijk op en zorgt er voor dat de applicatie crasht:

```
1 Unhandled Exception: System.DivideByZeroException: Division by zero
2   at Tests.RuntimeError.Main (System.String[] argv) [0x00000]
```

7.17 Tests/SyntaxIncorrect.vnvd

In deze test zijn er enkele syntax fouten opgenomen die de taal niet mag accepteren.

7.17.1 Testopzet

Deze test kan gecompileerd worden met de volgende aanroep:

```
1 Vnvd.exe SyntaxIncorrect.vnvd
```

Deze applicatie is een VNVD implementatie van testprogramma *A.2* in de practicumhandleiding.

7.17.2 Testresultaten

De VNVD compiler komt met de verwachte foutmelding bij een poging tot compilatie:

```
1  Vnvd Compiler v1.3.3.8
2  ======================
3
4  line 12:13 no viable alternative at input '*'
5  line 15:11 mismatched input ':' expecting SEMICOLON
6  line 18:13 missing LPAREN at 'gebin'
7  line 18:23 missing IN at 'repeat'
8  line 18:29 mismatched input ';' expecting RPAREN
```

8 Conclusies

Wij hebben veel tijd en werk gestoken in deze compiler en het eindresultaat is dan ook te zien. We hebben een volwaardige compiler gebouwd voor onze eigen gedefinieerde taal. Tijdens het ontwerpen en implementeren zijn we veel moeilijkheden tegengekomen, maar hebben deze stuk voor stuk kunnen oplossen. Het resultaat is dan ook een mooie compiler met veel functies.

De opdracht is op verscheidene niveau's leerzaam geweest. Naast basiskennis over het opbouwen van een compiler hebben we ook veel inzichten vergaard over het, op conceptueel, ontwerpen van een taal en haar controleer en compileer systeem. De uiteindelijke taal is immers opgebouwd uit bewezen concepten, met daar onze eigen invulling aan. Er zijn immers niet veel object georiënteerde expressie talen.

Ook het .NET framework waarop de taal gebouwd is heeft ons goed geholpen. Door te kijken hoe constructies waren uitgewerkt in onze "moedertaal", C#, konden we gebruik maken van de nuttige concepten. Daarnaast kregen we de mogelijkheid om te bedenken wat wij anders, en hopelijk beter, zouden doen en dit vervolgens ook uit te voeren.

Zeker het feit dat wij een expressietaal in een object georiënteerde mal hebben gegooid was in de eerste ontwikkel fase niet altijd even makkelijk. We kunnen nu echter succesvol concluderen dat het goed kan en enkele interessante taal mogelijkheden met zich meebrengt. Vooral het concept dat alles iets kan retourneren is een mooie insteek om mee te gaan programmeren. Dit geeft namelijk veel meer gevoel voor de achterliggende stack machine.

Al met al is VNVD het resultaat van veel werk en enthousiasme, waar we dan ook enige trots voor koesteren. Ons inziens is de taal zeker bruikbaar voor allerhande programmeer doeleinden. Of het nu grafische applicaties, servers, tools of libraries betreft, het is allemaal te maken en kost weinig moeite.

Indien we verder zouden willen gaan met VNVD zouden volgende stappen nog meer compatibiliteit met .NET standaarden, bootstrap en een IDE zijn. Waarschijnlijk zouden we ook enkele voor dit vak verplichte expressies, zoals de *read()* en de *write()*, verwijderen om meer in de geest, en abstractie, van de taal te blijven.

Er valt dus te concluderen dat VNVD een geslaagd en leuk project was, dat ook nog een uitermate bruikbaar product heeft opgeleverd.

A Compilatie van de VNVD compiler

Om de VNVD compiler zelf te compileren zijn er allereerst een aantal vereisten. Deze moeten zich op het systeem bevinden om de compilatie uit te kunnen voeren. Net zoals het nodig is om het *.NET Framework* te hebben om de compiler zelf te draaien en de programma's die deze produceert te executeren heb je dit framework ook nodig om de compiler te bouwen.

Het is overigens ook mogelijk om te werken met een open source variant van dit framework. In dat geval valt het aan te raden voor *Mono* te kiezen.

Daarnaast zijn natuurlijk de broncode zelf en de C# libraries van ANTLR vereist. Deze bevinden zich gelukkig allemaal in het meegeleverde pakket. De compilatie kan gedraaid worden vanuit de map *bin* die zich in de broncode map *Vnvd* bevindt. Het commando om te compileren is, in het geval men het *.NET Framework* gebruikt:

```
1 csc /out:Vnvd.exe /target:exe ..\AbstractHelper.cs ..\CheckerHelper
    .cs ..\CheckerManual.cs ..\EntryType.cs ..\GeneratorHelper.cs
    ..\GeneratorManual.cs ..\IdEntry.cs ..\ISymTab.cs ..\
    LibraryChecker.cs ..\Pair.cs ..\Parameter.cs ..\Qualifier.cs
    ..\StringHelper.cs ..\SymbolTable.cs ..\TreeNodeAdapter.cs ..\
    TreeNode.cs ..\Vnvd.cs ..\Antlr\VnvdChecker.cs ..\Antlr\
    VnvdGenerator.cs ..\Antlr\VnvdLexer.cs ..\Antlr\VnvdParser.cs
    ..\UserDefined\CustomBinder.cs ..\UserDefined\ErrorType.cs ..\
    UserDefined\IMethodOrConstructor.cs ..\UserDefined\MethodType.
    cs ..\UserDefined\NullType.cs ..\UserDefined\UserConstructor.cs
     ..\UserDefined\UserEnum.cs ..\UserDefined\UserField.cs ..\
    UserDefined\UserLocal.cs ..\UserDefined\UserMethod.cs ..\
    UserDefined\UserParameter.cs ..\UserDefined\UserType.cs ..\
    Properties\AssemblyInfo.cs /r:System.dll /r:System.Data.dll /r:
    System.Xml.dll /r:Antlr3.Runtime.dll /r:Antlr3.Utility.dll /r:
    antlr.runtime.dll /r:StringTemplate.dll
```

In het geval dat er met *Mono* gewerkt wordt is het commando:

```
1 gmcs2 ../AbstractHelper.cs ../CheckerHelper.cs ../CheckerManual.cs
    ../EntryType.cs ../GeneratorHelper.cs ../GeneratorManual.cs ../
    IdEntry.cs ../ISymTab.cs ../LibraryChecker.cs ../Pair.cs ../
    Parameter.cs ../Qualifier.cs ../StringHelper.cs ../SymbolTable.
    cs ../TreeNodeAdapter.cs ../TreeNode.cs ../Vnvd.cs ../Antlr/
    VnvdChecker.cs ../Antlr/VnvdGenerator.cs ../Antlr/VnvdLexer.cs
    ../Antlr/VnvdParser.cs ../UserDefined/CustomBinder.cs ../
    UserDefined/ErrorType.cs ../UserDefined/IMethodOrConstructor.cs
     ../UserDefined/MethodType.cs ../UserDefined/NullType.cs ../
    UserDefined/UserConstructor.cs ../UserDefined/UserEnum.cs ../
    UserDefined/UserField.cs ../UserDefined/UserLocal.cs ../
    UserDefined/UserMethod.cs ../UserDefined/UserParameter.cs ../
    UserDefined/UserType.cs ../Properties/AssemblyInfo.cs -r:System
    -r:System.Data -r:System.Xml -r:Antlr3.Runtime.dll -r:Antlr3.
    Utility.dll -r:antlr.runtime.dll -r:StringTemplate.dll -out:
    Vnvd.exe
```

Nadat de compilatie is uitgevoerd is er een bestand *Vnvd.exe* aangemaakt in de huidige map. Deze executable is de VNVD compiler.

B Gebruiksaanwijzing van de VNVD compiler

De VNVD compiler kan aangeroepen worden met een aantal bestanden (met een bestandsnaam eindigend op *.vnvd*) kan worden meegegeven. Deze komen met volledige naam achter de aanroep naar de compiler op de commando regel.

Met de VNVD compiler kunnen zowel executables als libraries gecompileerd worden. Voor het compileren van executables dient de bestandsnaam van de te genereren assembly te eindigen op *.exe*. Daarnaast dient deze één statische *Main()* of *Main(String[] args)* te kennen die *void* retouneert. Voor een library dient de bestandsnaam te eindigen op *.dll*.

De VNVD compiler kent een aantal compiler switches die nodig zijn voor het juist functioneren van de compiler. Deze kunnen voor de bestandsnamen worden toegevoegd aan de commando regel. Deze switches vallen te verdelen in de switches voor normaal gebruik en de switches die vanwege didactische redenen zijn toegevoegd. De normale switches zijn als volgt:

——**out:<executable_naam>** Hiermee kan men specificeren hoe de gecompileerde executable moet gaan heten.

——**ref:<assembly_naam>** Hiermee kan men een assembly ter referentie toevoegen. De klassen en methoden in deze assembly kunnen dan gebruikt worden door de linker. VNVD houdt hierbij rekening met systeem paden, zodat assemblies uit het .NET framework makkelijk kunnen worden toegevoegd.

De switches geïmplementeerd voor de ontwikkeling van de compiler en leerdoeleinden zijn als volgt:

——**ast** De compiler zal een bestand ast.txt aanmaken, met daarin een tekstuele representatie van de *abstract syntax tree*.

——**dot** De compiler zal een bestand ast.dot aanmaken, met daarin een grafische representatie van de *abstract syntax tree*.

——**nochecker** Hiermee kan de checker fase tijdelijk uit worden gezet.

——**nogenerator** Hiermee kan de code generator fase tijdelijk uit worden gezet.

Tot slot kent de compiler nog een switch die geen enkel doel dient, namelijk:

——**van** De compiler zal "Nee"naar de standard output schrijven en afsluiten.

Een voorbeeld aanroep naar VNVD ziet er als volgt uit:

```
1 Vnvd.exe --ref:System --ref:System.Windows.Forms --ref:System.
    Drawing --out:BKE.exe Bord.vnvd Mark.vnvd MensSpeler.vnvd Spel.
    vnvd ComputerSpeler.vnvd IStrategie.vnvd Speler.vnvd BkeGui.
    vnvd IMessageListener.vnvd BesteStrategie.vnvd DommeStrategie.
    vnvd
```

C ANTLR lexer en parser specificatie

Dit is de specificatie van zowel de lexer als de parser in ANTLR.

```
1  grammar Vnvd;
2
3  options
4  {
5      k = 1;
6      language = CSharp2;
7      output = AST;
8  }
9
10 tokens
11 {
12     COLON = ':';
13     SEMICOLON = ';';
14     LPAREN = '(';
15     RPAREN = ')';
16     COMMA = ',';
17     PERIOD = '.';
18     LCURLYBRACE = '{';
19     RCURLYBRACE = '}';
20     LBRACKET = '[';
21     RBRACKET = ']';
22     QUESTION = '?';
23     TILDE = '~';
24     DCOLON = '::';
25
26     ADDEVENT = '+=';
27     REMEVENT = '-=';
28
29     BECOMES = '=';
30     PLUS = '+';
31     MINUS = '-';
32     MULTIPLY = '*';
33     DIVISION = '/';
34
35     LE = '<';
36     LEQ = '<=';
37     GE = '>';
38     GEQ = '>=';
39     EQ = '==';
40     NEQ = '!=';
41
42     AND = '&&';
43     OR = '||';
44
45     LXOR = '^';
46     LAND = '&';
47     LOR = '|';
48
49     NOT = '!';
50     MOD = '%';
51
52     USING = 'import';
53     NAMESPACE = 'namespace';
54     CLASS = 'class';
55     INTERFACE = 'interface';
56     NEW = 'new';
57     STATIC = 'static';
58     VOID = 'void';
59     IF = 'if';
```

```
60      THEN = 'then';
61      ELSE = 'else';
62      WHILE = 'while';
63      FOR = 'for';
64      ENUM = 'enum';
65      IN = 'in';
66      RETURN = 'return';
67      CONST = 'const';
68      TRUE = 'true';
69      FALSE = 'false';
70      BASE = 'base';
71      THIS = 'this';
72      METHOD = '~method~';
73      IMETHOD = '~imethod~';
74      FIELD = '~field~';
75      CONSTRUCTOR = '~constructor~';
76      LOCAL = '~local~';
77      EXTENDS = 'extends';
78      IMPLEMENTS = 'implements';
79      FI  = 'fi';
80      PRINT = 'print';
81      READ = 'read';
82      NULL =  'null';
83      AS = 'as';
84      IS = 'is';
85      TRY = 'try';
86      CATCH = 'catch';
87      CAREBOX = 'carebox';
88      FINALLY = 'finally';
89      THROW = 'throw';
90
91      // functions
92      READ = 'read';
93      WRITE = 'write';
94
95      //modifiers
96      PRIVATE = 'private';
97      PUBLIC = 'public';
98      PROTECTED = 'protected';
99      INTERNAL = 'internal';
100     ABSTRACT = 'abstract';
101     STATIC = 'static';
102     VIRTUAL = 'virtual';
103     OVERRIDE = 'override';
104     INITONLY = 'initonly';
105
106     //imaginary tokens
107     PROGRAM = '~program~';
108     VARDECL = '~vardecl~';
109     PARAMETER = '~par~';
110     ARGUMENT = '~arg~';
111     SLIST = '~slist~';
112     ENUMOPTION = '~enumoption~';
113     CALL = '~call~';
114     FQUALIFIER = '~fqualifier~';
115     EQUALIFIER = '~equalifier~';
116     MODIFIERS = '~modifiers~';
117     EXPRBLOCK = '~expression~';
118     USAGE = '~usage~';
119     SCOPEEXPR = '~scopeepxr~';
120     CAST = '~cast~';
121     ARRAY = '~array~';
```

```
122     READVOID = '~readvoid~';
123     WRITEVOID = '~writevoid~';
124     ALIST = '~alist~';
125     NEWARR = '~newarr~';
126     NEWARRELEMS = '~newarrelems~';
127     ARRELEMS = '~arrelems~';
128     ARRELEM = '~arrelem~';
129     SCONSTRUCTOR = '~sconstructor~';
130     ICATCH = '~icatch~';
131 }
132
133 @namespace
134 {
135     Vnvd.Antlr
136 }
137
138 @lexer::namespace
139 {
140     Vnvd.Antlr
141 }
142
143 program
144     :   import_stat* namespace_decl* EOF
145     ->  ^(PROGRAM import_stat* namespace_decl*)
146     ;
147
148 import_stat
149     :   USING qualifier SEMICOLON
150     ->  ^(USING qualifier)
151     ;
152
153 namespace_decl
154     :   NAMESPACE qualifier LCURLYBRACE namespace_body RCURLYBRACE
155     ->  ^(NAMESPACE qualifier namespace_body?)
156     ;
157
158 namespace_body
159     :   (class_decl | interface_decl | enum_decl)*
160     ;
161
162 class_decl
163     :   CLASS mod=class_mod_list name=qualifier (EXTENDS base_=
            qualifier)? (IMPLEMENTS interface_=type_list)? class_body
164     ->  ^(CLASS ^(MODIFIERS $mod) $name ^(EXTENDS $base_)? ^(
            IMPLEMENTS $interface_)? class_body?)
165     ;
166
167 enum_decl
168     :   ENUM mod=access_modifier name=qualifier enum_body
169     ->  ^(ENUM ^(MODIFIERS $mod) $name enum_body?)
170     ;
171
172 enum_body
173     :   LCURLYBRACE (IDENTIFIER (COMMA IDENTIFIER)*)? RCURLYBRACE
174     ->  ^(ENUMOPTION IDENTIFIER)*
175     ;
176
177 interface_decl
178     :   INTERFACE mod=class_mod_list name=qualifier (EXTENDS base_=
            qualifier)? interface_body
179     ->  ^(INTERFACE ^(MODIFIERS $mod) $name ^(EXTENDS $base_)?
            interface_body?)
```

```
180     ;
181
182 interface_body
183     :   LCURLYBRACE interface_method* RCURLYBRACE
184     ->  interface_method*
185     ;
186
187 interface_method
188     :   type_qualifier IDENTIFIER LPAREN parameter_list RPAREN
            SEMICOLON
189     ->  ^(IMETHOD type_qualifier IDENTIFIER parameter_list)
190     ;
191
192 class_mod_list
193     :   access_modifier (ABSTRACT)?
194     ;
195
196 method_mod_list
197     : access_modifier (STATIC | virtual_modifier)?
198     ;
199
200 field_mod_list
201     :   access_modifier STATIC? INITONLY?
202     ;
203
204 constructor_mod_list
205     :   access_modifier
206     ;
207
208 abstract_mod_list
209     :   access_modifier ABSTRACT
210     ;
211
212 type_list
213     :   qualifier (COMMA! qualifier)*
214     ;
215
216 class_body
217     :   LCURLYBRACE! class_element* RCURLYBRACE!
218     ;
219
220 class_element
221     :   (method_mod_list type_qualifier IDENTIFIER LPAREN) =>
            method_decl
222     |   (constructor_mod_list IDENTIFIER LPAREN) =>
            constructor_decl
223     |   (abstract_mod_list) => abstract_method_decl
224     |   field_decl
225     |   static_constructor
226     ;
227
228 static_constructor
229     :   STATIC block_expression
230     ->  ^(SCONSTRUCTOR block_expression)
231     ;
232
233 literal_value
234     :   NUMBER
235     |   STRING
236     |   CHAR
237     |   TRUE
238     |   FALSE
```

```
239         |   NULL
240         ;
241
242     field_decl
243         :   mod=field_mod_list qual=qualifier IDENTIFIER (COMMA
                IDENTIFIER)* SEMICOLON
244         ->  ^(FIELD ^(MODIFIERS $mod) $qual IDENTIFIER)+
245         ;
246
247     method_decl
248         :   mod=method_mod_list type_qualifier IDENTIFIER LPAREN
                parameter_list RPAREN block_expression
249         ->  ^(METHOD ^(MODIFIERS $mod) type_qualifier IDENTIFIER
                parameter_list block_expression)
250         ;
251
252     abstract_method_decl
253         :   mod=abstract_mod_list type_qualifier IDENTIFIER LPAREN
                parameter_list RPAREN SEMICOLON
254         ->  ^(METHOD ^(MODIFIERS $mod) type_qualifier IDENTIFIER
                parameter_list)
255         ;
256
257     constructor_decl
258         :   mod=constructor_mod_list IDENTIFIER LPAREN parameter_list
                RPAREN (COLON BASE LPAREN argument_list RPAREN)?
                block_expression
259         ->  ^(CONSTRUCTOR ^(MODIFIERS $mod) IDENTIFIER parameter_list
                ^(BASE argument_list?)? block_expression)
260         ;
261
262     statement
263         :   (qualifier IDENTIFIER) => declaration_statement SEMICOLON!
264         |   const_decl_statement SEMICOLON!
265         |   expression SEMICOLON!
266         |   SEMICOLON!
267         |   control_statement
268         ;
269
270     control_statement
271         :   while_statement
272         |   for_statement
273         |   try_statement
274         |   throw_statement
275         ;
276
277     throw_statement
278         :   THROW expression SEMICOLON
279         ->  ^(THROW expression)
280         ;
281
282     try_statement
283         :   TRY block_expression (finally_block | (catch_block+
                finally_block?))
284         ->  ^(TRY block_expression catch_block* finally_block?)
285         ;
286
287     catch_block
288         :   (CATCH | CAREBOX) LPAREN qualifier IDENTIFIER RPAREN
                block_expression
289         ->  ^(ICATCH ^(LOCAL qualifier IDENTIFIER) block_expression)
290         ;
```

```
291
292 finally_block
293     :   FINALLY block_expression
294     ->  ^(FINALLY block_expression)
295     ;
296
297 while_statement
298     :   WHILE LPAREN scope_expr RPAREN do_=block_expression
299     ->  ^(WHILE scope_expr $do_)
300     ;
301
302 for_statement
303     :   FOR LPAREN type_=qualifier IDENTIFIER IN expression RPAREN
            do_=block_expression
304     ->  ^(FOR ^(LOCAL $type_ IDENTIFIER) expression $do_)
305     ;
306
307 scope_expr
308     :   statement+
309     -> ^(SCOPEEXPR statement+)
310     ;
311
312 declaration_statement
313     :   (qualifier IDENTIFIER BECOMES) => type_=qualifier
            IDENTIFIER BECOMES expression
314     ->  ^(LOCAL $type_ IDENTIFIER) ^(BECOMES ^(USAGE ^(FQUALIFIER
            IDENTIFIER)) expression)
315     |   type_=qualifier IDENTIFIER (COMMA IDENTIFIER)*
316     ->  ^(LOCAL $type_ IDENTIFIER)+
317     ;
318
319 const_decl_statement
320     :   CONST qualifier IDENTIFIER BECOMES const_value (COMMA
            IDENTIFIER BECOMES const_value)*
321     ->  ^(CONST qualifier IDENTIFIER const_value)+
322     ;
323
324 const_value
325     :   literal_value
326     |   LCURLYBRACE literal_value (COMMA literal_value)*
            RCURLYBRACE
327     ->  ^(ARRELEMS (^(ARRELEM literal_value))+)
328     ;
329
330 object_creation_expression
331     :   NEW qualifier LPAREN argument_list RPAREN
332     ->  ^(NEW qualifier argument_list?)
333     ;
334
335 array_creation_expression_elemented
336     :   NEW qualifier LCURLYBRACE expression (COMMA expression)*
            RCURLYBRACE
337     ->  ^(NEWARRELEMS qualifier ^(ARRELEMS (^(ARRELEM expression))
            *))
338     ;
339
340 array_creation_expression
341     :   NEW qualifier LBRACKET expression RBRACKET
342     ->  ^(NEWARR qualifier expression)
343     ;
344
345 expression
```

```
346     :   assignment_expression
347     ;
348
349 assignment_expression
350     :   t1=ternary_expression ((b=BECOMES^ | ADDEVENT^ | REMEVENT^)
             t2=assignment_expression)?
351     ;
352
353 ternary_expression
354     :   IF LPAREN scope_expr RPAREN THEN expression (ELSE
            expression)? FI
355     ->  ^(IF scope_expr expression expression?)
356     |   conditional_or_operator_expression
357     ;
358
359 conditional_or_operator_expression
360     :   conditional_and_operator_expression (OR^
            conditional_and_operator_expression)*
361     ;
362
363 conditional_and_operator_expression
364     :   logic_or_operator_expression (AND^
            logic_or_operator_expression)*
365     ;
366
367 logic_or_operator_expression
368     :   logic_xor_operator_expression (LOR^
            logic_xor_operator_expression)*
369     ;
370
371 logic_xor_operator_expression
372     :   logic_and_operator_expression (LXOR^
            logic_and_operator_expression)*
373     ;
374
375 logic_and_operator_expression
376     :   logic_expression (LAND^ logic_expression)*
377     ;
378
379 logic_expression
380     :   cast_as_expression ((LE^ | LEQ^ | GE^ | GEQ^ | EQ^ | NEQ^)
            cast_as_expression)*
381     ;
382
383 cast_as_expression
384     :   primary_expression ((AS^ | IS^) qualifier)*
385     ;
386
387 primary_expression
388     :   secundary_expression ((PLUS^ | MINUS^) secundary_expression
            )*
389     ;
390
391 secundary_expression
392     :   logicnotoperator_expression ((MULTIPLY^ | DIVISION^ | MOD^)
             logicnotoperator_expression )*
393     ;
394
395 logicnotoperator_expression
396     :   (NOT^ | PLUS^ | MINUS^)* cast_expression
397     ;
398
```

```
399 cast_expression
400     :   (LPAREN qualifier RPAREN cast_expression) => (LPAREN
            qualifier RPAREN cast_expression)
401     ->  ^(CAST qualifier cast_expression)
402     |   invocation_or_load_expression
403     ;
404
405 invocation_or_load_expression
406     :   operand ((DCOLON^ IDENTIFIER (LPAREN! argument_list RPAREN
            !)?) | (LBRACKET^ expression RBRACKET!))*
407     ;
408
409 operand
410     :   qualifier
411     ->  ^(USAGE qualifier)
412     |   NUMBER
413     |   FLOAT
414     |   STRING
415     |   CHAR
416     |   TRUE
417     |   FALSE
418     |   NULL
419     |   THIS
420     |   BASE
421     |   (NEW qualifier LPAREN) => object_creation_expression
422     |   (NEW qualifier LBRACKET) => array_creation_expression
423     |   (NEW qualifier LCURLYBRACE) =>
            array_creation_expression_elemented
424     |   (READ LPAREN IDENTIFIER RPAREN) => READ LPAREN IDENTIFIER
            RPAREN
425     ->  ^(READ IDENTIFIER)
426     |   READ LPAREN IDENTIFIER (COMMA IDENTIFIER)+ RPAREN
427     ->  ^(SCOPEEXPR ^(READVOID IDENTIFIER)+)
428     |   (WRITE LPAREN expression RPAREN) => WRITE LPAREN expression
             RPAREN
429     ->  ^(WRITE expression)
430     |   WRITE LPAREN expression (COMMA expression)+ RPAREN
431     ->  ^(SCOPEEXPR ^(WRITEVOID expression)+)
432     |   LPAREN expression RPAREN
433     ->  ^(EXPRBLOCK expression)
434     |   block_expression
435     ;
436
437 block_expression
438     :   LCURLYBRACE statement* RCURLYBRACE
439     ->  ^(SLIST statement*)
440     ;
441
442 parameter_list
443     :   (qualifier IDENTIFIER (COMMA qualifier IDENTIFIER)*)?
444     ->  ^(PARAMETER (qualifier IDENTIFIER)*)
445     ;
446
447 argument_list
448     :   (expression (COMMA expression)*)?
449     ->  ^(ALIST ^(ARGUMENT expression)*)
450     ;
451
452 type_qualifier
453     :   VOID
454     |   qualifier
455     ;
```

```
456
457 qualifier
458     :   (IDENTIFIER (PERIOD IDENTIFIER)* (LBRACKET RBRACKET)) => (
            IDENTIFIER (PERIOD IDENTIFIER)* (LBRACKET RBRACKET))
459     ->  ^(FQUALIFIER IDENTIFIER+ ARRAY)
460     |   IDENTIFIER (PERIOD IDENTIFIER)*
461     ->  ^(FQUALIFIER IDENTIFIER+)
462     ;
463
464 access_modifier
465     :   PRIVATE | PUBLIC | PROTECTED | INTERNAL
466     ;
467
468 virtual_modifier
469     :   VIRTUAL | OVERRIDE
470     ;
471
472
473 // LEXER
474 IDENTIFIER
475     :   (LETTER | '_') (LETTER | DIGIT | '_')*
476     ;
477
478 STRING
479     :   ('"' (QUOTED_CHARACTER | ~('"' | '\\'))* '"')
480     ;
481
482 CHAR
483     :   '\'' (QUOTED_CHARACTER | ~('\'' | '\\')) '\''
484     ;
485
486 QUOTED_CHARACTER
487     :   '\\' .
488     ;
489
490 NUMBER
491     :   (MINUS | PLUS)? DIGIT+
492     ;
493
494 FLOAT
495     :   (MINUS | PLUS)? DIGIT+ '.' DIGIT+
496     ;
497
498 COMMENT
499     :   '//' .* '\n'
500         { $channel=HIDDEN; }
501         | '/*' .* '*/'
502         { $channel=HIDDEN; }
503     ;
504
505 WS
506     :   (' ' | '\t' | '\f' | '\r' | '\n')+
507         { $channel=HIDDEN; }
508     ;
509
510 fragment DIGIT  :   ('0'..'9') ;
511 fragment LOWER  :   ('a'..'z') ;
512 fragment UPPER  :   ('A'..'Z') ;
513 fragment LETTER :   LOWER | UPPER ;
```

D ANTLR checker specificatie

Dit is de specificatie van de checker in ANTLR.

```
1  tree grammar VnvdChecker;
2
3  options
4  {
5      ASTLabelType = TreeNode;
6      tokenVocab = Vnvd;
7      language = CSharp2;
8  }
9
10 @header
11 {
12     using System.Collections.Generic;
13     using Vnvd;
14     using Vnvd.UserDefined;
15 }
16
17 @namespace
18 {
19     Vnvd.Antlr
20 }
21
22 @members
23 {
24     private CheckerHelper h;
25     public int SemanticErrorCount { get; private set; }
26     public String Filename { get; set; }
27 }
28
29 @rulecatch
30 {
31     catch (RecognitionException re)
32     {
33         ReportError(re);
34         Recover(input,re);
35     }
36     catch (CheckerException ex)
37     {
38         if (ex.GetLastError() != null)
39         {
40             ex.GetLastError().First.Entry = ErrorType.DefaultEntry;
41             ex.GetLastError().First.ReturnType = ErrorType.Default;
42         }
43         Console.WriteLine(Filename + "\t" + ex.ToString());
44         SemanticErrorCount++;
45     }
46 }
47
48 program
49     @init { this.h = new CheckerHelper(this); }
50     :   ^(node=PROGRAM import_stat* { h.BeginProgram(node); }
           namespace_decl*)
51         {
52             h.Program(node);
53         }
54     ;
55
56 import_stat
57     :   ^(node=USING qual=qualifier)
58         {
```

```
59              h.Import(node, $qual.ids);
60          }
61      ;
62
63  namespace_decl
64      :   ^(node=NAMESPACE qual=qualifier { h.BeginNamespace(node,
            $qual.ids); } (class_decl | interface_decl | enum_decl)*)
65          {
66              h.Namespace(node, $qual.ids);
67          }
68      ;
69
70
71  class_decl
72      :   ^(node=CLASS ^(mod=MODIFIERS modifier*) qual=qualifier (^(
            ext=EXTENDS qualifier))? (^(imp=IMPLEMENTS qualifier*))?
73          { h.BeginClass(node, h.Modifiers(mod), $qual.ids, h.
                Extendation(ext), h.Implements(imp)); } class_element*)
74          {
75              h.Class(node, h.Modifiers(mod), $qual.ids, h.
                    Extendation(ext), h.Implements(imp));
76          }
77      ;
78
79  enum_decl
80      :   ^(node=ENUM ^(mod=MODIFIERS modifier*) qual=qualifier (^(
            ENUMOPTION IDENTIFIER))*)
81      ;
82
83  interface_decl
84      :   ^(node=INTERFACE ^(mod=MODIFIERS modifier*) qual=qualifier
            (^(ext=EXTENDS qualifier))?
85          { h.BeginInterface(node, h.Modifiers(mod), $qual.ids, h.
                Extendation(ext)); } interface_method*)
86          {
87              h.Interface(node, h.Modifiers(mod), $qual.ids, h.
                    Extendation(ext));
88          }
89      ;
90
91  interface_method
92      :   ^(node=IMETHOD qual=type_qualifier id=IDENTIFIER ^(par=
            PARAMETER (qualifier IDENTIFIER)*))
93          {
94              h.InterfaceMethod(node, $qual.ids, id.Text, h.
                    Parameters(par));
95          }
96      ;
97
98  class_element
99      :   method_decl
100     |   constructor_decl
101     |   field_decl
102     |   static_constructor
103     ;
104
105 literal_value
106     :   node=NUMBER
107         {
108             h.LiteralNumber(node);
109         }
110     |   node=STRING
```

```
111             {
112                 h.LiteralString(node);
113             }
114         |   node=FLOAT
115             {
116                 h.LiteralFloat(node);
117             }
118         |   node=CHAR
119             {
120                 h.LiteralCharacter(node);
121             }
122         |   node=TRUE
123             {
124                 h.LiteralBoolean(node);
125             }
126         |   node=FALSE
127             {
128                 h.LiteralBoolean(node);
129             }
130         ;
131
132     abstract_mod_list
133         :   ^(MODIFIERS access_modifier ABSTRACT)
134         ;
135
136     access_modifier
137         :   PRIVATE | PUBLIC | PROTECTED | INTERNAL
138         ;
139
140     field_decl
141         :   ^(node=FIELD ^(mod=MODIFIERS modifier*) qual=qualifier id=
                IDENTIFIER)
142             {
143                 h.Field(node, h.Modifiers(mod), $qual.ids, id.Text);
144             }
145         ;
146
147     method_decl
148         :   ^(node=METHOD ^(mod=MODIFIERS modifier*) return_=
                type_qualifier id=IDENTIFIER ^(par=PARAMETER (qualifier
                IDENTIFIER)*) { h.BeginMethod(node, h.Modifiers(mod),
                $return_.ids, id.Text, h.Parameters(par));}
                block_expression?)
149             {
150                 h.Method(node, h.Modifiers(mod), $return_.ids, id.Text,
                        h.Parameters(par));
151             }
152         ;
153
154     constructor_decl
155         :   ^(node=CONSTRUCTOR ^(mod=MODIFIERS modifier*) id=IDENTIFIER
                ^(par=PARAMETER (qualifier IDENTIFIER)*) { h.
                BeginConstructorBeforeBaseCall(node); } (^(bas=BASE (^(
                ALIST argument*))))? { h.BeginConstructor(node, h.Modifiers
                (mod), h.Parameters(par), bas); } block_expression)
156             {
157                 h.Constructor(node, h.Modifiers(mod), h.Parameters(par)
                        );
158             }
159         ;
160
161     static_constructor
```

```
162     :   ^(node=SCONSTRUCTOR { h.BeginStaticConstructor(node); }
            block_expression)
163     {
164         h.EndStaticConstructor(node);
165     }
166     ;
167
168 statement
169     :   expression
170     |   declaration_statement
171     |   const_decl_statement
172     |   control_statement
173     ;
174
175 control_statement
176     :   while_statement
177     |   for_statement
178     |   try_statement
179     |   throw_statement
180     ;
181
182 throw_statement
183     :   ^(node=THROW expression)
184         {
185             h.ThrowStatement(node);
186         }
187     ;
188
189 try_statement
190     :   ^(node=TRY { h.BeginTry(node); } block_expression
            catch_block* finally_block?)
191         {
192             h.EndTry(node);
193         }
194     ;
195
196 catch_block
197     @init { h.OpenScope(); }
198     :   ^(node=CATCH ^(node2=LOCAL qual=qualifier id=IDENTIFIER) {
             h.DeclarationStatement(node2, $qual.ids, id.Text); h.
            BeginCatch(node, node2, $qual.ids, id.Text); }
            block_expression)
199         {
200             h.EndCatch(node, $qual.ids, id.Text);
201             h.CloseScope();
202         }
203     ;
204
205 finally_block
206     :   ^(node=FINALLY { h.BeginFinally(node); } block_expression)
207         {
208             h.EndFinally(node);
209         }
210     ;
211
212 while_statement
213     @init { h.OpenScope(); }
214     :   ^(node=WHILE scope_expr block_expression)
215         {
216             h.WhileLoop(node);
217
218             h.CloseScope();
```

```
219             }
220         ;
221
222     for_statement
223         @init { h.OpenScope(); }
224         :   ^(node=FOR ^(node2=LOCAL qual=qualifier id=IDENTIFIER)
                expression { h.DeclarationStatement(node2, $qual.ids, id.
                Text); h.BeginFor(node, node2, $qual.ids, id.Text); }
                block_expression)
225             {
226                 h.ForLoop(node);
227
228                 h.CloseScope();
229             }
230         ;
231
232     scope_expr
233         :   ^(node=SCOPEEXPR statement+)
234         {
235             h.ScopeExpr(node);
236         }
237         ;
238
239     declaration_statement
240         :   ^(node=LOCAL qual=qualifier id=IDENTIFIER)
241             {
242                 h.DeclarationStatement(node, $qual.ids, id.Text);
243             }
244         ;
245
246     const_decl_statement
247         :   ^(node=CONST qual=qualifier id=IDENTIFIER const_value)
248             {
249                 h.ConstantDeclaration(node, $qual.ids, id.Text);
250             }
251         ;
252
253     const_value
254         :   literal_value
255         |   ^(node=ARRELEMS (const_array_element)+)
256             {
257                 h.ConstantArray(node);
258             }
259         ;
260
261     const_array_element
262         :   ^(node=ARRELEM literal_value)
263             {
264                 h.ConstantArrayElement(node);
265             }
266         ;
267
268     expression
269         :   assignment_expression
270         ;
271
272     assignment_expression
273         :   ternary_expression
274         |   ^(node=BECOMES assignment_expression assignment_expression)
275             {
276                 h.AssignmentExpr(node);
277             }
```

```
278     |   ^(node=ADDEVENT assignment_expression assignment_expression
            )
279         {
280             h.AddEventExpr(node);
281         }
282     |   ^(node=REMEVENT assignment_expression assignment_expression
            )
283         {
284             h.RemoveEventExpr(node);
285         }
286     ;
287
288 ternary_expression
289     :   { h.OpenScope(); }
290         ^(node=IF scope_expr expression expression?)
291         {
292             h.TernaryExpr(node);
293             h.CloseScope();
294         }
295     |   conditional_or_operator_expression
296     ;
297
298 conditional_or_operator_expression
299     :   conditional_and_operator_expression
300     |   ^(node=OR conditional_or_operator_expression
            conditional_or_operator_expression)
301         {
302             h.EndConditionalOrExpr(node);
303         }
304     ;
305
306 conditional_and_operator_expression
307     :   logic_or_operator_expression
308     |   ^(node=AND conditional_and_operator_expression
            conditional_and_operator_expression)
309         {
310             h.EndConditionalAndExpr(node);
311         }
312     ;
313
314 logic_or_operator_expression
315     :   logic_xor_operator_expression
316     |   ^(node=LOR logic_or_operator_expression
            logic_or_operator_expression)
317         {
318             h.LogicOrExpr(node);
319         }
320     ;
321
322 logic_xor_operator_expression
323     :   logic_and_operator_expression
324     |   ^(node=LXOR logic_xor_operator_expression
            logic_xor_operator_expression)
325         {
326             h.LogicXorExpr(node);
327         }
328     ;
329
330 logic_and_operator_expression
331     :   logic_expression
332     |   ^(node=LAND logic_and_operator_expression
            logic_and_operator_expression)
```

```
333             {
334                 h.LogicAndExpr(node);
335             }
336         ;
337 
338     logic_expression
339         :   cast_as_expression
340         |   ^(node=LE primary_expression primary_expression)
341             {
342                 h.LessExpr(node);
343             }
344         |   ^(node=LEQ primary_expression primary_expression)
345             {
346                 h.LessEqualExpr(node);
347             }
348         |   ^(node=GE primary_expression primary_expression)
349             {
350                 h.GreaterExpr(node);
351             }
352         |   ^(node=GEQ primary_expression primary_expression)
353             {
354                 h.GreaterEqualExpr(node);
355             }
356         |   ^(node=EQ primary_expression primary_expression)
357             {
358                 h.EqualExpr(node);
359             }
360         |   ^(node=NEQ primary_expression primary_expression)
361             {
362                 h.NotEqualExpr(node);
363             }
364         ;
365 
366     cast_as_expression
367         :   ^(node=AS cast_as_expression qual=qualifier)
368             {
369                 h.CastAsExpression(node, $qual.ids);
370             }
371         |   ^(node=IS cast_as_expression qual=qualifier)
372             {
373                 h.IsTypeExpression(node, $qual.ids);
374             }
375         |   primary_expression
376         ;
377 
378     primary_expression
379         :   (^(PLUS primary_expression primary_expression)) => ^(node=
                PLUS primary_expression primary_expression)
380             {
381                 h.PlusExpr(node);
382             }
383         |   (^(MINUS primary_expression primary_expression)) => ^(node=
                MINUS primary_expression primary_expression)
384             {
385                 h.MinusExpr(node);
386             }
387         |   secundary_expression
388         ;
389 
390     secundary_expression
391         :   ^(node=MULTIPLY secundary_expression secundary_expression)
392             {
```

```
393             h.MultiplyExpr(node);
394         }
395     |   ^(node=DIVISION secundary_expression secundary_expression)
396         {
397             h.DivisionExpr(node);
398         }
399     |   ^(node=MOD secundary_expression secundary_expression)
400         {
401             h.ModExpr(node);
402         }
403     |   logicnotoperator_expression
404     ;
405
406 logicnotoperator_expression
407     :   ^(node=NOT logicnotoperator_expression)
408         {
409             h.LogicNotExpr(node);
410         }
411     |   ^(node=PLUS logicnotoperator_expression)
412         {
413             h.UnaryPlusExpr(node);
414         }
415     |   ^(node=MINUS logicnotoperator_expression)
416         {
417             h.UnaryMinusExpr(node);
418         }
419     |   cast_expression
420     ;
421
422 cast_expression
423     :   ^(node=CAST qual=qualifier cast_expression)
424     {
425         h.CastExpression(node, $qual.ids);
426     }
427     |   invocation_or_load_expression
428     ;
429
430 invocation_or_load_expression
431     :   ^(node=DCOLON invocation_or_load_expression id=IDENTIFIER
        (^(ALIST argument*))?)
432         {
433             h.InvocationExpr(node, id.Text);
434         }
435     |   ^(node=LBRACKET invocation_or_load_expression expression)
436         {
437             h.AccessArray(node);
438         }
439     |   operand
440     ;
441
442 operand
443     :   ^(node=USAGE qual=qualifier)
444         {
445             h.VariableUsed(node, $qual.ids);
446         }
447     |   literal_value
448     |   node=THIS
449         {
450             h.LiteralThis(node);
451         }
452     |   node=BASE
453         {
```

```
454             h.LiteralBase(node);
455         }
456     |   node=NULL
457         {
458             h.LiteralNull(node);
459         }
460     |   read_write_expression
461     |   object_creation_expression
462     |   array_creation_expression
463     |   ^(node=EXPRBLOCK expression)
464         {
465             h.ExpressionBlock(node);
466         }
467     |   block_expression
468     |   scope_expr
469     ;
470
471 read_write_expression
472     :   ^(node=READ qual=IDENTIFIER)
473         {
474             h.ReadExpression(node, qual.Text);
475         }
476     |   ^(node=READVOID qual=IDENTIFIER)
477         {
478             h.ReadVoidExpression(node, qual.Text);
479         }
480     |   ^(node=WRITE expression)
481         {
482             h.WriteExpression(node);
483         }
484     |   ^(node=WRITEVOID expression)
485         {
486             h.WriteVoidExpression(node);
487         }
488     ;
489
490 array_creation_expression
491     :   ^(node=NEWARR qualifier expression)
492         {
493             h.ArrayCreationExpr(node);
494         }
495     |   ^(node=NEWARRELEMS qualifier  ^(ARRELEMS
        array_initial_elements*))
496         {
497             h.ArrayElementedCreationExpr(node);
498         }
499     ;
500
501 array_initial_elements
502     :   ^(node=ARRELEM { h.BeforeArrayElementedElement(node); }
        expression)
503         {
504             h.ArrayElementedElement(node);
505         }
506     ;
507
508 object_creation_expression
509     :   ^(node=NEW qual=qualifier (^(ALIST argument*))?)
510         {
511             h.ObjectCreation(node, $qual.ids);
512         }
513     ;
```

```
514
515 block_expression
516     :   { h.OpenScope(); } ^(node=SLIST statement*)
517         {
518             h.StatementBlock(node);
519
520             h.CloseScope();
521         }
522     ;
523
524 argument
525     :   ^(ARGUMENT expression)
526     ;
527
528 qualifier returns [Qualifier ids]
529     :   ^(fqual=FQUALIFIER IDENTIFIER+ ARRAY?)
530         {
531             $ids = h.FullQualifier(fqual);
532         }
533     ;
534
535 type_qualifier returns [Qualifier ids]
536     :   v=VOID
537         {
538             $ids = new Qualifier() { "System", "Void" };
539         }
540     |   qual=qualifier
541         {
542             $ids = h.TypeQualifier($qual.ids);
543         }
544     ;
545
546 modifier
547     :   PRIVATE | PUBLIC | PROTECTED | INTERNAL | ABSTRACT | STATIC
          | VIRTUAL | OVERRIDE | INITONLY
548     ;
```

E ANTLR code generator specificatie

Dit is de specificatie van de code generator in ANTLR.

```
1  tree grammar VnvdGenerator;
2
3  options
4  {
5      tokenVocab = Vnvd;
6      language = CSharp2;
7      ASTLabelType = TreeNode;
8  }
9
10 @header
11 {
12     using System.Collections.Generic;
13     using Vnvd;
14 }
15
16 @namespace
17 {
18     Vnvd.Antlr
19 }
20
21 @members
22 {
23     private GeneratorHelper h = new GeneratorHelper();
24 }
25
26 program
27     :   ^(node=PROGRAM import_stat* { h.BeginProgram(node); }
          namespace_decl*)
28         {
29             h.Program(node);
30         }
31     ;
32
33 import_stat
34     :   ^(node=USING qual=qualifier)
35         {
36             h.Import(node, qual);
37         }
38     ;
39
40 namespace_decl
41     :   ^(node=NAMESPACE qual=qualifier { h.BeginNamespace(node,
          qual); } (class_decl | interface_decl | enum_decl)*)
42         {
43             h.Namespace(node, qual);
44         }
45     ;
46
47
48 class_decl
49     :   ^(node=CLASS ^(mod=MODIFIERS modifier*) qual=qualifier (^(
          ext=EXTENDS qualifier))? (^(imp=IMPLEMENTS qualifier*))?
50         { h.BeginClass(node, h.Modifiers(mod), qual, h.Extendation(
              ext), h.Implements(imp)); } class_element*)
51         {
52             h.Class(node, h.Modifiers(mod), qual, h.Extendation(ext
                  ), h.Implements(imp));
53         }
54     ;
```

```
55
56  enum_decl
57      :   ^(node=ENUM ^(mod=MODIFIERS modifier*) qual=qualifier (^(
            ENUMOPTION IDENTIFIER))*)
58      ;
59
60  interface_decl
61      :   ^(node=INTERFACE ^(mod=MODIFIERS modifier*) qual=qualifier
            (^(ext=EXTENDS qualifier))?
62          { h.BeginInterface(node, h.Modifiers(mod), qual, h.
                Extendation(ext)); } interface_method*)
63          {
64              h.Interface(node, h.Modifiers(mod), qual, h.Extendation
                    (ext));
65          }
66      ;
67
68  interface_method
69      :   ^(node=IMETHOD qual=type_qualifier id=IDENTIFIER ^(par=
            PARAMETER (qualifier IDENTIFIER)*))
70          {
71              h.InterfaceMethod(node, qual, id.Text, h.Parameters(par
                    ));
72          }
73      ;
74
75  class_element
76      :   method_decl
77      |   constructor_decl
78      |   field_decl
79      |   static_constructor
80      ;
81
82  literal_value
83      :   node=NUMBER
84          {
85              h.LiteralNumber(node);
86          }
87      |   node=FLOAT
88          {
89              h.LiteralFloat(node);
90          }
91      |   node=STRING
92          {
93              h.LiteralString(node);
94          }
95      |   node=CHAR
96          {
97              h.LiteralCharacter(node);
98          }
99      |   node=TRUE
100         {
101             h.LiteralBoolean(node);
102         }
103     |   node=FALSE
104         {
105             h.LiteralBoolean(node);
106         }
107     ;
108
109 constant_literal_value
110     :   NUMBER
```

```
111         |   STRING
112         |   FLOAT
113         |   CHAR
114         |   TRUE
115         |   FALSE
116         ;
117
118     abstract_mod_list
119         :   ^(MODIFIERS access_modifier ABSTRACT)
120         ;
121
122     access_modifier
123         :   PRIVATE | PUBLIC | PROTECTED | INTERNAL
124         ;
125
126     field_decl
127         :   ^(node=FIELD ^(mod=MODIFIERS modifier*) qual=qualifier id=
                IDENTIFIER)
128             {
129                 h.Field(node, h.Modifiers(mod), qual, id.Text);
130             }
131         ;
132
133     method_decl
134         :   ^(node=METHOD ^(mod=MODIFIERS modifier*) return_=
                type_qualifier id=IDENTIFIER ^(par=PARAMETER (qualifier
                IDENTIFIER)*) { h.BeginMethod(node, h.Modifiers(mod),
                return_, id.Text, h.Parameters(par));} block_expression?)
135             {
136                 h.Method(node, h.Modifiers(mod), return_, id.Text, h.
                        Parameters(par));
137             }
138         ;
139
140     constructor_decl
141         :   ^(node=CONSTRUCTOR ^(mod=MODIFIERS modifier*) id=IDENTIFIER
                 ^(par=PARAMETER (qualifier IDENTIFIER)*) { h.
                BeginConstructorBeforeBaseCall(node); } (^(bas=BASE ^(ALIST
                 argument*)))? { h.BeginConstructor(node, h.Modifiers(mod),
                 h.Parameters(par), bas); } block_expression)
142             {
143                 h.Constructor(node, h.Modifiers(mod), h.Parameters(par)
                        );
144             }
145         ;
146
147     static_constructor
148         :   ^(node=SCONSTRUCTOR { h.BeginStaticConstructor(node); }
                block_expression)
149         {
150             h.EndStaticConstructor(node);
151         }
152         ;
153
154     statement
155         :   expression
156         |   declaration_statement
157         |   const_decl_statement
158         |   control_statement
159         ;
160
161     control_statement
```

```
162     :   while_statement
163     |   for_statement
164     |   try_statement
165     |   throw_statement
166     ;
167
168 throw_statement
169     :   ^(node=THROW expression)
170         {
171             h.ThrowStatement(node);
172         }
173     ;
174
175 try_statement
176     :   ^(node=TRY { h.BeginTry(node); } block_expression
            catch_block* finally_block?)
177         {
178             h.EndTry(node);
179         }
180     ;
181
182 catch_block
183     @init { h.OpenScope(); }
184     :   ^(node=CATCH ^(node2=LOCAL qual=qualifier id=IDENTIFIER) {
             h.DeclarationStatement(node2, qual, id.Text); h.BeginCatch
            (node, node2, qual, id.Text); } block_expression)
185         {
186             h.EndCatch(node, qual, id.Text);
187             h.CloseScope();
188         }
189     ;
190
191 finally_block
192     :   ^(node=FINALLY { h.BeginFinally(node); } block_expression)
193         {
194             h.EndFinally(node);
195         }
196     ;
197
198 while_statement
199     @init { h.OpenScope(); }
200     :   ^(node=WHILE { h.WhileStart(node); } scope_expr { h.
            WhileAfterExpr(node); } block_expression)
201         {
202             h.WhileLoop(node);
203             h.CloseScope();
204         }
205     ;
206
207 for_statement
208     @init { h.OpenScope(); }
209     :   ^(node=FOR ^(node2=LOCAL qual=qualifier id=IDENTIFIER)
            expression { h.DeclarationStatement(node2, qual, id.Text);
            h.BeginFor(node, node2, qual, id.Text); } block_expression)
210         {
211             h.ForLoop(node);
212
213             h.CloseScope();
214         }
215     ;
216
217 scope_expr
```

```
218     :   ^(node=SCOPEEXPR statement+)
219     {
220         h.ScopeExpr(node);
221     }
222     ;
223
224 declaration_statement
225     :   ^(node=LOCAL qual=qualifier id=IDENTIFIER)
226         {
227             h.DeclarationStatement(node, qual, id.Text);
228         }
229     ;
230
231 const_decl_statement
232     :   ^(node=CONST qual=qualifier id=IDENTIFIER const_value)
233         {
234             h.ConstantDeclaration(node, qual, id.Text);
235         }
236     ;
237
238 const_value
239     :   constant_literal_value
240     |   ^(node=ARRELEMS (const_array_element)+)
241     ;
242
243 const_array_element
244     :   ^(node=ARRELEM constant_literal_value)
245     ;
246
247 expression
248     :   assignment_expression
249     ;
250
251 assignment_expression
252     :   ternary_expression
253     |   ^(node=BECOMES assignment_expression assignment_expression)
254         {
255             h.AssignmentExpr(node);
256         }
257     |   ^(node=ADDEVENT assignment_expression assignment_expression
            )
258         {
259             h.AddEventExpr(node);
260         }
261     |   ^(node=REMEVENT assignment_expression assignment_expression
            )
262         {
263             h.RemoveEventExpr(node);
264         }
265     ;
266
267 ternary_expression
268     :   { h.OpenScope(); }
269         ^(node=IF scope_expr { h.TernaryThen(node); } expression {
                h.TernaryElse(node); } expression?)
270         {
271             h.TernaryExpr(node);
272             h.CloseScope();
273         }
274     |   conditional_or_operator_expression
275     ;
276
```

```
277 conditional_or_operator_expression
278     :   conditional_and_operator_expression
279     |   ^(node=OR { h.BeginCondOr(node); }
            conditional_or_operator_expression { h.MiddleCondOr(node);
            } conditional_or_operator_expression)
280         {
281             h.EndConditionalOrExpr(node);
282         }
283     ;
284
285 conditional_and_operator_expression
286     :   logic_or_operator_expression
287     |   ^(node=AND { h.BeginCondAnd(node); }
            conditional_and_operator_expression { h.MiddleCondAnd(node)
            ; } conditional_and_operator_expression)
288         {
289             h.EndConditionalAndExpr(node);
290         }
291     ;
292
293 logic_or_operator_expression
294     :   logic_xor_operator_expression
295     |   ^(node=LOR logic_or_operator_expression
            logic_or_operator_expression)
296         {
297             h.LogicOrExpr(node);
298         }
299     ;
300
301 logic_xor_operator_expression
302     :   logic_and_operator_expression
303     |   ^(node=LXOR logic_xor_operator_expression
            logic_xor_operator_expression)
304         {
305             h.LogicXorExpr(node);
306         }
307     ;
308
309 logic_and_operator_expression
310     :   logic_expression
311     |   ^(node=LAND logic_and_operator_expression
            logic_and_operator_expression)
312         {
313             h.LogicAndExpr(node);
314         }
315     ;
316
317 logic_expression
318     :   cast_as_expression
319     |   ^(node=LE logic_expression logic_expression)
320         {
321             h.LessExpr(node);
322         }
323     |   ^(node=LEQ logic_expression logic_expression)
324         {
325             h.LessEqualExpr(node);
326         }
327     |   ^(node=GE logic_expression logic_expression)
328         {
329             h.GreaterExpr(node);
330         }
331     |   ^(node=GEQ logic_expression logic_expression)
```

```
332             {
333                 h.GreaterEqualExpr(node);
334             }
335         |   ^(node=EQ logic_expression logic_expression)
336             {
337                 h.EqualExpr(node);
338             }
339         |   ^(node=NEQ logic_expression logic_expression)
340             {
341                 h.NotEqualExpr(node);
342             }
343         ;
344
345     cast_as_expression
346         :   ^(node=AS cast_as_expression qual=qualifier)
347             {
348                 h.CastAsExpression(node, $qual.ids);
349             }
350         |   ^(node=IS cast_as_expression qual=qualifier)
351             {
352                 h.IsTypeExpression(node, $qual.ids);
353             }
354         |   primary_expression
355         ;
356
357     primary_expression
358         :
359             (^(PLUS primary_expression primary_expression)) => ^(node=
                    PLUS primary_expression primary_expression)
360             {
361                 h.PlusExpr(node);
362             }
363         |   (^(MINUS primary_expression primary_expression)) => ^(node=
                MINUS primary_expression primary_expression)
364             {
365                 h.MinusExpr(node);
366             }
367         |   secundary_expression
368         ;
369
370     secundary_expression
371         :   logicnotoperator_expression
372         |   ^(node=MULTIPLY secundary_expression secundary_expression)
373             {
374                 h.MultiplyExpr(node);
375             }
376         |   ^(node=DIVISION secundary_expression secundary_expression)
377             {
378                 h.DivisionExpr(node);
379             }
380         |   ^(node=MOD secundary_expression secundary_expression)
381             {
382                 h.ModExpr(node);
383             }
384         ;
385
386     logicnotoperator_expression
387         :   ^(node=NOT logicnotoperator_expression)
388             {
389                 h.LogicNotExpr(node);
390             }
391         |   ^(node=PLUS logicnotoperator_expression)
```

```
392             {
393                 h.UnaryPlusExpr(node);
394             }
395         |   ^(node=MINUS logicnotoperator_expression)
396             {
397                 h.UnaryMinusExpr(node);
398             }
399         |   cast_expression
400         ;
401
402     cast_expression
403         :   ^(node=CAST qual=qualifier cast_expression)
404         {
405             h.CastExpression(node, qual);
406         }
407         |   invocation_or_load_expression
408         ;
409
410     invocation_or_load_expression
411         :   ^(node=DCOLON invocation_or_load_expression id=IDENTIFIER
                (^(ALIST argument*))?)
412             {
413                 h.InvocationExpr(node, id.Text);
414             }
415         |   ^(node=LBRACKET invocation_or_load_expression expression)
416             {
417                 h.AccessArray(node);
418             }
419         |   operand
420         ;
421
422     operand
423         :   ^(node=USAGE qual=qualifier)
424             {
425                 h.VariableUsed(node, qual);
426             }
427         |   literal_value
428         |   node=THIS
429             {
430                 h.LiteralThis(node);
431             }
432         |   node=BASE
433             {
434                 h.LiteralBase(node);
435             }
436         |   node=NULL
437             {
438                 h.LiteralNull(node);
439             }
440         |   read_write_expression
441         |   object_creation_expression
442         |   array_creation_expression
443         |   ^(node=EXPRBLOCK expression)
444             {
445                 h.ExpressionBlock(node);
446             }
447         |   block_expression
448         |   scope_expr
449         ;
450
451     read_write_expression
452         :   ^(node=READ qual=IDENTIFIER)
```

```
453             {
454                 h.ReadExpression(node, qual.Text);
455             }
456         |   ^(node=READVOID qual=IDENTIFIER)
457             {
458                 h.ReadVoidExpression(node, qual.Text);
459             }
460         |   ^(node=WRITE expression)
461             {
462                 h.WriteExpression(node);
463             }
464         |   ^(node=WRITEVOID expression)
465             {
466                 h.WriteVoidExpression(node);
467             }
468         ;
469
470     object_creation_expression
471         :   ^(node=NEW qual=qualifier (^(ALIST argument*))?)
472             {
473                 h.ObjectCreation(node, qual);
474             }
475         ;
476
477     array_creation_expression
478         :   ^(node=NEWARR qualifier expression)
479             {
480                 h.ArrayCreationExpr(node);
481             }
482         |   ^(node=NEWARRELEMS qualifier { h.
                BeforeArrayElementedCreationExpr(node); } ^(ARRELEMS
                array_initial_elements*))
483             {
484                 h.ArrayElementedCreationExpr(node);
485             }
486         ;
487
488     array_initial_elements
489         :   ^(node=ARRELEM { h.BeforeArrayElementedElement(node); }
                expression)
490             {
491                 h.ArrayElementedElement(node);
492             }
493         ;
494
495     block_expression
496         :   { h.OpenScope(); } ^(node=SLIST statement*)
497             {
498                 h.StatementBlock(node);
499                 h.CloseScope();
500             }
501         ;
502
503     argument
504         :   ^(ARGUMENT expression)
505         ;
506
507     qualifier returns [Qualifier ids]
508         :   ^(fqual=FQUALIFIER IDENTIFIER+ ARRAY?)
509             {
510                 ids = h.FullQualifier(fqual);
511             }
```

```
512     ;
513
514 type_qualifier returns [Qualifier ids]
515     :   v=VOID
516         {
517             ids = new Qualifier() { "System", "Void" };
518         }
519     |   qual=qualifier
520         {
521             ids = h.TypeQualifier(qual);
522         }
523     ;
524
525 modifier
526     :   PRIVATE | PUBLIC | PROTECTED | INTERNAL | ABSTRACT | STATIC
              | VIRTUAL | OVERRIDE | INITONLY
527     ;
```

F Uitgebreid testprogramma

In deze sectie volgt de volledige broncode en gebruikshandleiding van het uitgebreide testprogramma. Dit testprogramma is een implementatie van het bekende spelletje *Boter, Kaas en Eieren*. Het bijbehorende testplan en de testresultaten zijn in het verslag zelf terug te vinden.

F.1 Broncode

De broncode van het uitgebreide testprogramma, *Boter, Kaas en Eieren*, bestaat uit verscheidene bestanden. Deze staan hieronder op alfabetische volgorde gegeven.

F.1.1 BesteStrategie.vnvd

De broncode van *BesteStrategie.vnvd* is als volgt:

```
1  import System;
2  import System.Collections;
3  import System.Drawing;
4  import System.Timers;
5  import System.Windows.Forms;
6  import System.ComponentModel;
7
8  namespace BoterKaasEieren
9  {
10     class public BesteStrategie implements IStrategie
11     {
12        private Mark dezeSpeler;
13        private Random random;
14
15        public BesteStrategie()
16        {
17           this::random = new Random();
18        }
19
20        public Int32 BerekenZet(Bord bord, Mark mark)
21        {
22           this::dezeSpeler = mark;
23           this::_berekenZet(bord, mark)::zet;
24        }
25
26        private Zet _berekenZet(Bord bord, Mark mark)
27        {
28           Boolean cont = true;
29           Zet besteZet = null;
30           Int32 count = 0;
31           Int32 offset = this::random::Next(Bord::DIM * Bord::DIM);
32           while (count < Bord::DIM * Bord::DIM && cont;)
33           {
34              Int32 i = (count + offset) % (Bord::DIM * Bord::DIM);
35              if (bord::IsLeegVakje(i);) then
36              {
37                 Zet zet = new Zet();
38                 zet::zet = i;
39
40                 bord::SetVakje(i, mark);
41                 if (bord::GameOver();) then
42                 {
43                    if (bord::HeeftGewonnen(mark);) then
```

```
44                      {
45                          cont = false;
46                          zet::waarde = 1;
47                      }
48                      else
49                      {
50                          zet::waarde = 0;
51                      } fi;
52                  }
53                  else
54                  {
55                      zet::waarde = -this::_berekenZet(bord, mark::
                            Other())::waarde;
56                  } fi;
57
58                  if (besteZet == null || zet::waarde > besteZet::
                        waarde;) then
59                  {
60                      besteZet = zet;
61                  } fi;
62                  bord::SetVakje(i, Mark::EMPTY);
63              } fi;
64              count = count + 1;
65          }
66          besteZet;
67      }
68  }
69
70  class public Zet
71  {
72      public Int32 zet;
73      public Int32 waarde;
74  }
75  }
```

F.1.2 BkeGui.vnvd

De broncode van *BkeGui.vnvd* is als volgt:

```
1  import System;
2  import System.Collections;
3  import System.Drawing;
4  import System.Timers;
5  import System.Windows.Forms;
6  import System.Drawing;
7  import System.Threading;
8
9  namespace BoterKaasEieren
10 {
11     class public BkeGui extends Form implements IMessageListener
12     {
13         private Spel spel;
14
15         private Button button;
16         private Button[] tiles;
17         private Label label;
18
19         private RadioButton s1radio1, s1radio2, s1radio3;
20         private RadioButton s2radio1, s2radio2, s2radio3;
21
22         public static void Main()
23         {
```

```
24              Application::Run(new BkeGui());
25          }
26
27          public BkeGui()
28          {
29              this::set_Text("Boter Kaas Eieren");
30            this::set_Size(new Size(350, 300));
31
32            Panel panel = new Panel();
33            panel::set_Size(new Size(80, 80));
34            panel::set_Location(new Point(10, 10));
35            this::get_Controls()::Add(panel);
36
37            Label l = new Label();
38            l::set_Text("Speler 1");
39            panel::get_Controls()::Add(l);
40
41            this::s1radio1 = new RadioButton();
42            this::s1radio1::set_Text("Human");
43            this::s1radio1::set_Checked(true);
44            this::s1radio1::set_Location(new Point(0, 20));
45            panel::get_Controls()::Add(this::s1radio1);
46
47            this::s1radio2 = new RadioButton();
48            this::s1radio2::set_Text("Domme AI");
49            this::s1radio2::set_Location(new Point(0, 40));
50            panel::get_Controls()::Add(this::s1radio2);
51
52            this::s1radio3 = new RadioButton();
53            this::s1radio3::set_Text("Slimme AI");
54            this::s1radio3::set_Location(new Point(0, 60));
55            panel::get_Controls()::Add(this::s1radio3);
56
57            panel = new Panel();
58            panel::set_Size(new Size(80, 80));
59            panel::set_Location(new Point(100, 10));
60            this::get_Controls()::Add(panel);
61
62            l = new Label();
63            l::set_Text("Speler 2");
64            panel::get_Controls()::Add(l);
65
66            this::s2radio1 = new RadioButton();
67            this::s2radio1::set_Text("Human");
68            this::s2radio1::set_Checked(true);
69            this::s2radio1::set_Location(new Point(0, 20));
70            panel::get_Controls()::Add(this::s2radio1);
71
72            this::s2radio2 = new RadioButton();
73            this::s2radio2::set_Text("Domme AI");
74            this::s2radio2::set_Location(new Point(0, 40));
75            panel::get_Controls()::Add(this::s2radio2);
76
77            this::s2radio3 = new RadioButton();
78            this::s2radio3::set_Text("Slimme AI");
79            this::s2radio3::set_Location(new Point(0, 60));
80            panel::get_Controls()::Add(this::s2radio3);
81
82            panel = new Panel();
83            panel::set_Size(new Size(250, 250));
84            panel::set_Location(new Point(25, 110));
85            this::get_Controls()::Add(panel);
```

```
86
87              this::tiles = new Button[9];
88              Int32 i = 0;
89              while (i < 9;)
90              {
91                  Button b = new Button();
92                  this::tiles[i] = b;
93                  b::set_Name(i::ToString());
94                  b::set_Text(i::ToString());
95                  b::set_Size(new Size(40, 40));
96                  b::set_Location(new Point((i % 3) * 40, (i / 3) * 40))
                        ;
97                  b::set_Enabled(false);
98                  b::Click += new EventHandler(this::move_Clicked);
99                  panel::get_Controls()::Add(b);
100                 i = i + 1;
101             }
102             this::button = new Button();
103             this::button::set_Location(new Point(180, 20));
104             this::button::set_Text("Start!");
105             this::button::Click += new EventHandler(this::
                    button_Clicked);
106             this::get_Controls()::Add(this::button);
107
108             this::label = new Label();
109             this::label::set_Text("Press the start button!");
110             this::label::set_Location(new Point(180, 50));
111             this::label::set_AutoSize(true);
112             this::get_Controls()::Add(this::label);
113         }
114
115         public void MessageReceived(Object message)
116         {
117             if (message == null;) then
118             {
119                 Int32 i = 0;
120                 while (i < 9;)
121                 {
122                     this::tiles[i]::set_Text(i::ToString());
123                     this::tiles[i]::set_Enabled(true);
124                     i = i + 1;
125                 }
126                 Speler s = this::spel::GetHuidigeSpeler();
127                 this::label::set_Text(s::GetNaam() + " is aan zet!");
128             }
129             else if (message is Bord;) then
130             {
131                 Bord b = (Bord)message;
132                 Int32 i = 0;
133                 while (i < 9;)
134                 {
135                     this::tiles[i]::set_Text(b::GetVakje(i)::ToString()
                            );
136                     i = i + 1;
137                 }
138                 Speler s = this::spel::GetHuidigeSpeler();
139                 this::label::set_Text(s::GetNaam() + " is aan zet!");
140             }
141             else
142             {
143                 Int32 i = (Int32)message;
144                 if (i == -1;) then
```

```
145                 {
146                     this::label::set_Text("Gelijkspel!");
147                 }
148                 else
149                 {
150                     Speler s = this::spel::GetSpelers()[i];
151                     this::label::set_Text(s::GetNaam() + " heeft
                        gewonnen!");
152                 } fi;
153                 i = 0;
154                 while (i < 9;)
155                 {
156                     this::tiles[i]::set_Enabled(false);
157                     i = i + 1;
158                 }
159             } fi fi;
160         }
161
162         private void button_Clicked(Object sender, EventArgs e)
163         {
164             Speler s1, s2;
165             s1 = if (this::s1radio1::get_Checked();) then
166                 (Speler)new MensSpeler("Speler 1", Mark::XX)
167             else if (this::s1radio2::get_Checked();) then
168                 (Speler)new ComputerSpeler("Speler 1", Mark::XX, new
                    DommeStrategie())
169             else
170                 (Speler)new ComputerSpeler("Speler 1", Mark::XX, new
                    BesteStrategie())
171             fi fi;
172
173             s2 = if (this::s2radio1::get_Checked();) then
174                 (Speler)new MensSpeler("Speler 2", Mark::OO)
175             else if (this::s2radio2::get_Checked();) then
176                 (Speler)new ComputerSpeler("Speler 2", Mark::OO, new
                    DommeStrategie())
177             else
178                 (Speler)new ComputerSpeler("Speler 2", Mark::OO, new
                    BesteStrategie())
179             fi fi;
180
181           this::spel = new Spel(s1, s2);
182           this::spel::AddObserver(this);
183           Thread t = new Thread(new ThreadStart(this::spel::Play));
184           t::set_IsBackground(true);
185           t::Start();
186         }
187
188         private void move_Clicked(Object sender, EventArgs e)
189         {
190             if (this::spel::GetHuidigeSpeler() is MensSpeler;) then
191             {
192               Button b = (Button)sender;
193               Int32 move = Int32::Parse(b::get_Name());
194               MensSpeler s = (MensSpeler)this::spel::GetHuidigeSpeler
                    ();
195               Monitor::Enter(s);
196               s::SetTempMove(move);
197               Monitor::Pulse(s);
198               Monitor::Exit(s);
199             } fi;
200
```

```
201         }
202     }
203 }
```

F.1.3 Bord.vnvd

De broncode van *Bord.vnvd* is als volgt:

```
1   import System;
2
3   namespace BoterKaasEieren
4   {
5      class public Bord
6      {
7         public static initonly Int32 DIM;
8
9         private Mark[] bord;
10
11        static
12        {
13           Bord::DIM = 3;
14        }
15
16        public Bord()
17        {
18           this::bord = new Mark[Bord::DIM * Bord::DIM];
19           this::Reset();
20        }
21
22        public Bord DeepCopy()
23        {
24           Bord copy;
25           copy = new Bord();
26           Int32 i;
27           i = 0;
28           while (i < Bord::DIM * Bord::DIM;)
29           {
30              copy::SetVakje(i, this::GetVakje(i));
31           }
32           copy;
33        }
34
35        public Int32 Index(Int32 rij, Int32 kol)
36        {
37           Bord::DIM * rij + kol;
38        }
39
40        public void Reset()
41        {
42           Int32 i;
43           i = 0;
44           while (i < Bord::DIM * Bord::DIM;)
45           {
46              this::SetVakje(i, Mark::EMPTY);
47              i = i + 1;
48           }
49        }
50
51        public void SetVakje(Int32 rij, Int32 kol, Mark m)
52        {
53           this::SetVakje(this::Index(rij, kol), m);
54        }
```

```
55
56          public void SetVakje(Int32 i, Mark m)
57          {
58              this::bord[i] = m;
59          }
60
61          public Mark GetVakje(Int32 rij, Int32 kol)
62          {
63              this::GetVakje(this::Index(rij, kol));
64          }
65
66          public Mark GetVakje(Int32 i)
67          {
68              this::bord[i];
69          }
70
71          public Boolean HeeftRij(Mark mark)
72          {
73              Boolean found = false;
74            Int32 i = 0;
75            Int32 j;
76            while (i < Bord::DIM && !found;)
77            {
78                j = 0;
79                found = true;
80                while (j < Bord::DIM && found;)
81                {
82                    found = this::GetVakje(i, j) == mark;
83                    j = j + 1;
84                }
85                i = i + 1;
86            }
87            found;
88          }
89
90          public Boolean HeeftKolom(Mark mark)
91          {
92              Boolean found = false;
93            Int32 i = 0;
94            Int32 j;
95            while (i < Bord::DIM && !found;)
96            {
97                j = 0;
98                found = true;
99                while (j < Bord::DIM && found;)
100               {
101                   found = this::GetVakje(j, i) == mark;
102                   j = j + 1;
103               }
104               i = i + 1;
105           }
106           found;
107         }
108
109         public Boolean HeeftDiagonaal(Mark mark)
110         {
111           Boolean d1, d2;
112           Int32 i = 0;
113           d1 = d2 = true;
114           while (i < Bord::DIM && (d1 || d2);)
115           {
116               d1 = d1 && this::GetVakje(i, i) == mark;
```

```
117             d2 = d2 && this::GetVakje(i, Bord::DIM - i - 1) == mark
                    ;
118             i = i + 1;
119           }
120           d1 || d2;
121         }
122
123         public Boolean HeeftWinnaar()
124         {
125           this::HeeftGewonnen(Mark::XX) || this::HeeftGewonnen(Mark
                  ::OO);
126         }
127
128         public Boolean GameOver()
129         {
130           this::HeeftWinnaar() || this::IsVol();
131         }
132
133         public Boolean IsVol()
134         {
135           Int32 i;
136           i = 0;
137           Boolean vol;
138           vol = true;
139           while (i < Bord::DIM * Bord::DIM && vol;)
140           {
141             vol = this::GetVakje(i) != Mark::EMPTY;
142             i = i + 1;
143           }
144           vol;
145         }
146
147         public Boolean HeeftGewonnen(Mark m)
148         {
149           this::HeeftRij(m) || this::HeeftKolom(m) || this::
                  HeeftDiagonaal(m);
150         }
151
152         public Boolean IsVakje(Int32 rij, Int32 kolom)
153         {
154           this::IsVakje(this::Index(rij, kolom));
155         }
156
157         public Boolean IsVakje(Int32 i)
158         {
159           i < Bord::DIM * Bord::DIM && i >= 0;
160         }
161
162         public Boolean IsLeegVakje(Int32 rij, Int32 kolom)
163         {
164           this::IsLeegVakje(this::Index(rij, kolom));
165         }
166
167         public Boolean IsLeegVakje(Int32 i)
168         {
169           this::bord[i] == Mark::EMPTY;
170         }
171
172         public Boolean IsCorrecteZet(Int32 i)
173         {
174           this::IsVakje(i) && this::IsLeegVakje(i);
175         }
```

```
176
177         public Boolean IsCorrecteZet(Int32 rij, Int32 kol)
178         {
179             this::IsCorrecteZet(this::Index(rij, kol));
180         }
181     }
182 }
```

F.1.4 ComputerSpeler.vnvd

De broncode van *ComputerSpeler.vnvd* is als volgt:

```
1  import System;
2  import System.Threading;
3
4  namespace BoterKaasEieren
5  {
6      class public ComputerSpeler extends Speler
7      {
8          private IStrategie strategie;
9
10         public ComputerSpeler(String naam, Mark mark, IStrategie
               strategie) : base(naam, mark)
11         {
12             this::strategie = strategie;
13         }
14
15         public override Int32 BepaalZet(Bord bord)
16         {
17             this::strategie::BerekenZet(bord, this::GetMark());
18         }
19
20         public IStrategie GetStrategie()
21         {
22             this::strategie;
23         }
24
25         public void SetStrategie(IStrategie strategie)
26         {
27             this::strategie = strategie;
28         }
29     }
30 }
```

F.1.5 DommeStrategie.vnvd

De broncode van *DommeStrategie.vnvd* is als volgt:

```
1  import System;
2  import System.Collections;
3  import System.Drawing;
4  import System.Timers;
5  import System.Windows.Forms;
6  import System.ComponentModel;
7
8  namespace BoterKaasEieren
9  {
10     class public DommeStrategie implements IStrategie
11     {
12         private Random random;
13
```

```
14          public DommeStrategie()
15          {
16              this::random = new Random();
17          }
18
19          public Int32 BerekenZet(Bord bord, Mark mark)
20          {
21              ArrayList zetten = new ArrayList();
22              Int32 i = 0;
23              while (i < Bord::DIM * Bord::DIM;)
24              {
25                  if (bord::IsLeegVakje(i);) then
26                  {
27                      zetten::Add((Object)i);
28                  } fi;
29                  i = i + 1;
30              }
31              (Int32)zetten::get_Item(this::random::Next(0, zetten::
                    get_Count()));
32          }
33      }
34
35  }
```

F.1.6 IMessageListener.vnvd

De broncode van *IMessageListener.vnvd* is als volgt:

```
1   import System;
2   import System.Collections;
3   import System.Drawing;
4   import System.Timers;
5   import System.Windows.Forms;
6   import System.ComponentModel;
7
8   namespace BoterKaasEieren
9   {
10      interface public IMessageListener extends ISynchronizeInvoke
11      {
12          void MessageReceived(Object message);
13      }
14
15  }
```

F.1.7 IStrategie.vnvd

De broncode van *IStrategie.vnvd* is als volgt:

```
1   import System;
2   import System.Collections;
3   import System.Drawing;
4   import System.Timers;
5   import System.Windows.Forms;
6   import System.ComponentModel;
7
8   namespace BoterKaasEieren
9   {
10      interface public IStrategie
11      {
12          Int32 BerekenZet(Bord bord, Mark mark);
13      }
```

```
14
15  }
```

F.1.8 Mark.vnvd

De broncode van *Mark.vnvd* is als volgt:

```
1   import System;
2
3   namespace BoterKaasEieren
4   {
5       class public Mark
6       {
7           public static initonly Mark XX, OO, EMPTY;
8
9           private String sign;
10
11          private Mark(String sign)
12          {
13              this::sign = sign;
14          }
15
16          public Mark Other()
17          {
18              if (this == Mark::XX;) then Mark::OO else Mark::XX fi;
19          }
20
21          public override String ToString()
22          {
23              this::sign;
24          }
25
26          static
27          {
28              Mark::XX = new Mark("X");
29              Mark::OO = new Mark("O");
30              Mark::EMPTY = new Mark(" ");
31          }
32      }
33  }
```

F.1.9 MensSpeler.vnvd

De broncode van *MensSpeler.vnvd* is als volgt:

```
1   import System;
2   import System.Threading;
3
4   namespace BoterKaasEieren
5   {
6       class public MensSpeler extends Speler
7       {
8           private Int32 tempMove;
9
10          public MensSpeler(String naam, Mark mark) : base(naam, mark)
11          {
12              this::tempMove = -1;
13          }
14
15          public override Int32 BepaalZet(Bord bord)
16          {
```

```
17            Monitor::Enter(this);
18            while (!bord::IsCorrecteZet(this::tempMove);)
19            {
20               Monitor::Wait(this);
21            }
22
23            Monitor::Exit(this);
24            Int32 ret = this::tempMove;
25            this::tempMove = -1;
26            ret;
27         }
28
29         public Int32 GetTempMove()
30         {
31            this::tempMove;
32         }
33
34         public void SetTempMove(Int32 tempMove)
35         {
36            this::tempMove = tempMove;
37         }
38      }
39   }
```

F.1.10 Speler.vnvd

De broncode van *Speler.vnvd* is als volgt:

```
1   import System;
2
3   namespace BoterKaasEieren
4   {
5      class public abstract Speler
6      {
7         private String naam;
8         private Mark mark;
9
10        protected Speler(String naam, Mark mark)
11        {
12           this::naam = naam;
13           this::mark = mark;
14        }
15
16        public void DoeZet(Bord bord)
17        {
18           Int32 i = this::BepaalZet(bord);
19           bord::SetVakje(i, this::mark);
20        }
21
22        public String GetNaam()
23        {
24           this::naam;
25        }
26
27        public Mark GetMark()
28        {
29           this::mark;
30        }
31
32        public abstract Int32 BepaalZet(Bord bord);
33     }
34  }
```

F.1.11 Spel.vnvd

De broncode van *Spel.vnvd* is als volgt:

```
1  import System;
2  import System.Collections;
3  import System.Threading;
4
5  namespace BoterKaasEieren
6  {
7     class public Spel
8     {
9        public static initonly Int32 AANTAL_SPELERS;
10
11       private Speler[] spelers;
12       private Int32 huidig;
13       private Bord bord;
14       private ArrayList listeners;
15
16       public Spel(Speler s1, Speler s2)
17       {
18          this::spelers = new Speler[] { s1, s2 };
19          this::listeners = new ArrayList();
20          this::bord = new Bord();
21       }
22
23       public Bord GetBord()
24       {
25          this::bord;
26       }
27
28       public Int32 GetHuidig()
29       {
30          this::huidig;
31       }
32
33       public Speler GetHuidigeSpeler()
34       {
35          this::spelers[this::huidig];
36       }
37
38       public Speler[] GetSpelers()
39       {
40          this::spelers;
41       }
42
43       public void Reset()
44       {
45          this::bord::Reset();
46          this::huidig = 0;
47       }
48
49       public void Play()
50       {
51          this::Update(null);
52          while (!this::bord::GameOver();)
53          {
54             this::spelers[this::huidig]::DoeZet(this::bord);
55             this::huidig = (this::huidig + 1) % Spel::
                   AANTAL_SPELERS;
56
57             this::Update(this::bord);
58          }
```

```
59            this::Update(if (this::bord::HeeftWinnaar();) then (Object
                  )((this::huidig + 1) % Spel::AANTAL_SPELERS) else (
                  Object)-1 fi);
60        }
61
62        public void AddObserver(IMessageListener l)
63        {
64            this::listeners::Add(l);
65        }
66
67        public void RemoveObserver(IMessageListener l)
68        {
69            this::listeners::Remove(l);
70        }
71
72        public void Update(Object msg)
73        {
74            for (IMessageListener l in this::listeners)
75            {
76                if (l::get_InvokeRequired();) then
77                {
78                 l::Invoke(new ParameterizedThreadStart(l::
                      MessageReceived), new Object[] { msg });
79             }
80             else
81             {
82                   l::MessageReceived(msg);
83                }
84                fi;
85            }
86        }
87
88        static
89        {
90            Spel::AANTAL_SPELERS = 2;
91        }
92    }
93 }
```

F.2 Assembly

Tijdens de compilatie produceert de VNVD compiler de volgende assembly:

```
1  .assembly extern mscorlib
2  {
3    .ver 2:0:0:0
4    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .z\V.4..
5  }
6  .assembly extern System
7  {
8    .ver 2:0:0:0
9    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .z\V.4..
10 }
11 .assembly extern System.Windows.Forms
12 {
13   .ver 2:0:0:0
14   .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .z\V.4..
15 }
16 .assembly extern System.Drawing
17 {
18   .ver 2:0:0:0
19   .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A ) // .?_....:
```

```
20 }
21 .assembly 'BKE'
22 {
23   .hash algorithm 0x00008004
24   .ver  0:0:0:0
25 }
26 .module BKE // GUID = {78DF7BDF-F79A-430E-ABD2-45D230FAC7DE}
27
28
29 .namespace BoterKaasEieren
30 {
31   .class public auto ansi Bord
32     extends [mscorlib]System.Object
33   {
34     .field  public static  initonly  int32 DIM
35     .field  private  class BoterKaasEieren.Mark[] bord
36
37     // method line 1
38     .method public static  specialname  rtspecialname
39            default void '.cctor' ()  cil managed
40     {
41         // Method begins at RVA 0x20ec
42     // Code size 12 (0xc)
43     .maxstack 1
44     .locals init (
45         object  V_0)
46     IL_0000:  nop
47     IL_0001:  ldc.i4 3
48     IL_0006:  stsfld int32 BoterKaasEieren.Bord::DIM
49     IL_000b:  ret
50     } // end of method Bord::.cctor
51
52     // method line 2
53     .method public hidebysig  specialname  rtspecialname
54            instance default void '.ctor' ()  cil managed
55     {
56         // Method begins at RVA 0x2104
57     // Code size 35 (0x23)
58     .maxstack 6
59     .locals init (
60         object  V_0)
61     IL_0000:  ldarg.0
62     IL_0001:  call instance void object::'.ctor'()
63     IL_0006:  ldarg.0
64     IL_0007:  ldsfld int32 BoterKaasEieren.Bord::DIM
65     IL_000c:  ldsfld int32 BoterKaasEieren.Bord::DIM
66     IL_0011:  mul
67     IL_0012:  newarr BoterKaasEieren.Mark
68     IL_0017:  stfld class BoterKaasEieren.Mark[] BoterKaasEieren.
         Bord::bord
69     IL_001c:  ldarg.0
70     IL_001d:  call instance void class BoterKaasEieren.Bord::Reset
         ()
71     IL_0022:  ret
72     } // end of method Bord::.ctor
73
74     // method line 3
75     .method public hidebysig
76            instance default class BoterKaasEieren.Bord DeepCopy ()
                 cil managed
77     {
78         // Method begins at RVA 0x2134
```

```
79      // Code size 53 (0x35)
80      .maxstack 6
81      .locals init (
82          object  V_0,
83          class BoterKaasEieren.Bord  V_1,
84          int32   V_2)
85      IL_0000:  nop
86      IL_0001:  newobj instance void class BoterKaasEieren.Bord::'.
            ctor'()
87      IL_0006:  stloc.1
88      IL_0007:  ldc.i4 0
89      IL_000c:  stloc.2
90      IL_000d:  ldloc.2
91      IL_000e:  ldsfld int32 BoterKaasEieren.Bord::DIM
92      IL_0013:  ldsfld int32 BoterKaasEieren.Bord::DIM
93      IL_0018:  mul
94      IL_0019:  clt
95      IL_001b:  brfalse IL_0033
96
97      IL_0020:  ldloc.1
98      IL_0021:  ldloc.2
99      IL_0022:  ldarg.0
100     IL_0023:  ldloc.2
101     IL_0024:  call instance class BoterKaasEieren.Mark class
            BoterKaasEieren.Bord::GetVakje(int32)
102     IL_0029:  call instance void class BoterKaasEieren.Bord::
            SetVakje(int32, class BoterKaasEieren.Mark)
103     IL_002e:  br IL_000d
104
105     IL_0033:  ldloc.1
106     IL_0034:  ret
107     } // end of method Bord::DeepCopy
108
109     // method line 4
110     .method public hidebysig
111           instance default int32 Index (int32 rij, int32 kol)  cil
                  managed
112     {
113         // Method begins at RVA 0x2178
114     // Code size 17 (0x11)
115     .maxstack 3
116     .locals init (
117         object  V_0)
118     IL_0000:  nop
119     IL_0001:  ldsfld int32 BoterKaasEieren.Bord::DIM
120     IL_0006:  ldarg 1
121
122     IL_000a:  mul
123     IL_000b:  ldarg 2
124
125     IL_000f:  add
126     IL_0010:  ret
127     } // end of method Bord::Index
128
129     // method line 5
130     .method public hidebysig
131           instance default void Reset ()  cil managed
132     {
133         // Method begins at RVA 0x2198
134     // Code size 52 (0x34)
135     .maxstack 5
136     .locals init (
```

```
137           object  V_0,
138           int32   V_1)
139       IL_0000:  nop
140       IL_0001:  ldc.i4 0
141       IL_0006:  stloc.1
142       IL_0007:  ldloc.1
143       IL_0008:  ldsfld int32 BoterKaasEieren.Bord::DIM
144       IL_000d:  ldsfld int32 BoterKaasEieren.Bord::DIM
145       IL_0012:  mul
146       IL_0013:  clt
147       IL_0015:  brfalse IL_0033
148
149       IL_001a:  ldarg.0
150       IL_001b:  ldloc.1
151       IL_001c:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::EMPTY
152       IL_0021:  call instance void class BoterKaasEieren.Bord::
              SetVakje(int32, class BoterKaasEieren.Mark)
153       IL_0026:  ldloc.1
154       IL_0027:  ldc.i4 1
155       IL_002c:  add
156       IL_002d:  stloc.1
157       IL_002e:  br IL_0007
158
159       IL_0033:  ret
160       } // end of method Bord::Reset
161
162       // method line 6
163       .method public hidebysig
164             instance default void SetVakje (int32 rij, int32 kol,
                  class BoterKaasEieren.Mark m)  cil managed
165       {
166           // Method begins at RVA 0x21d8
167       // Code size 26 (0x1a)
168       .maxstack 6
169       .locals init (
170           object  V_0)
171       IL_0000:  nop
172       IL_0001:  ldarg.0
173       IL_0002:  ldarg.0
174       IL_0003:  ldarg 1
175
176       IL_0007:  ldarg 2
177
178       IL_000b:  call instance int32 class BoterKaasEieren.Bord::Index
              (int32, int32)
179       IL_0010:  ldarg 3
180
181       IL_0014:  call instance void class BoterKaasEieren.Bord::
              SetVakje(int32, class BoterKaasEieren.Mark)
182       IL_0019:  ret
183       } // end of method Bord::SetVakje
184
185       // method line 7
186       .method public hidebysig
187             instance default void SetVakje (int32 i, class
                  BoterKaasEieren.Mark m)  cil managed
188       {
189           // Method begins at RVA 0x2200
190       // Code size 21 (0x15)
191       .maxstack 3
192       .locals init (
```

```
193         object  V_0)
194     IL_0000:  nop
195     IL_0001:  ldarg.0
196     IL_0002:  ldfld class BoterKaasEieren.Mark[] BoterKaasEieren.
            Bord::bord
197     IL_0007:  ldarg 1
198
199     IL_000b:  ldarg 2
200
201     IL_000f:  stelem.any BoterKaasEieren.Mark
202     IL_0014:  ret
203     } // end of method Bord::SetVakje
204
205     // method line 8
206     .method public hidebysig
207             instance default class BoterKaasEieren.Mark GetVakje (
                    int32 rij, int32 kol)  cil managed
208     {
209         // Method begins at RVA 0x2224
210     // Code size 22 (0x16)
211     .maxstack 5
212     .locals init (
213         object  V_0)
214     IL_0000:  nop
215     IL_0001:  ldarg.0
216     IL_0002:  ldarg.0
217     IL_0003:  ldarg 1
218
219     IL_0007:  ldarg 2
220
221     IL_000b:  call instance int32 class BoterKaasEieren.Bord::Index
            (int32, int32)
222     IL_0010:  call instance class BoterKaasEieren.Mark class
            BoterKaasEieren.Bord::GetVakje(int32)
223     IL_0015:  ret
224     } // end of method Bord::GetVakje
225
226     // method line 9
227     .method public hidebysig
228             instance default class BoterKaasEieren.Mark GetVakje (
                    int32 i)  cil managed
229     {
230         // Method begins at RVA 0x2248
231     // Code size 17 (0x11)
232     .maxstack 3
233     .locals init (
234         object  V_0)
235     IL_0000:  nop
236     IL_0001:  ldarg.0
237     IL_0002:  ldfld class BoterKaasEieren.Mark[] BoterKaasEieren.
            Bord::bord
238     IL_0007:  ldarg 1
239
240     IL_000b:  ldelem.any BoterKaasEieren.Mark
241     IL_0010:  ret
242     } // end of method Bord::GetVakje
243
244     // method line 10
245     .method public hidebysig
246             instance default bool HeeftRij (class BoterKaasEieren.
                    Mark mark)  cil managed
247     {
```

```
248         // Method begins at RVA 0x2268
249     // Code size 113 (0x71)
250     .maxstack 7
251     .locals init (
252         object  V_0,
253         bool    V_1,
254         int32   V_2,
255         int32   V_3)
256     IL_0000:  nop
257     IL_0001:  ldc.i4.0
258     IL_0002:  stloc.1
259     IL_0003:  ldc.i4 0
260     IL_0008:  stloc.2
261     IL_0009:  ldloc.2
262     IL_000a:  ldsfld int32 BoterKaasEieren.Bord::DIM
263     IL_000f:  clt
264     IL_0011:  brfalse IL_001f
265
266     IL_0016:  ldloc.1
267     IL_0017:  ldc.i4.0
268     IL_0018:  ceq
269     IL_001a:  br IL_0020
270
271     IL_001f:  ldc.i4.0
272     IL_0020:  brfalse IL_006f
273
274     IL_0025:  ldc.i4 0
275     IL_002a:  stloc.3
276     IL_002b:  ldc.i4.1
277     IL_002c:  stloc.1
278     IL_002d:  ldloc.3
279     IL_002e:  ldsfld int32 BoterKaasEieren.Bord::DIM
280     IL_0033:  clt
281     IL_0035:  brfalse IL_0040
282
283     IL_003a:  ldloc.1
284     IL_003b:  br IL_0041
285
286     IL_0040:  ldc.i4.0
287     IL_0041:  brfalse IL_0062
288
289     IL_0046:  ldarg.0
290     IL_0047:  ldloc.2
291     IL_0048:  ldloc.3
292     IL_0049:  call instance class BoterKaasEieren.Mark class
            BoterKaasEieren.Bord::GetVakje(int32, int32)
293     IL_004e:  ldarg 1
294
295     IL_0052:  ceq
296     IL_0054:  stloc.1
297     IL_0055:  ldloc.3
298     IL_0056:  ldc.i4 1
299     IL_005b:  add
300     IL_005c:  stloc.3
301     IL_005d:  br IL_002d
302
303     IL_0062:  ldloc.2
304     IL_0063:  ldc.i4 1
305     IL_0068:  add
306     IL_0069:  stloc.2
307     IL_006a:  br IL_0009
308
```

```
309     IL_006f:  ldloc.1
310     IL_0070:  ret
311     } // end of method Bord::HeeftRij
312
313     // method line 11
314     .method public hidebysig
315             instance default bool HeeftKolom (class BoterKaasEieren.
                    Mark mark)  cil managed
316     {
317         // Method begins at RVA 0x22e8
318     // Code size 113 (0x71)
319     .maxstack 7
320     .locals init (
321         object  V_0,
322         bool    V_1,
323         int32   V_2,
324         int32   V_3)
325     IL_0000:  nop
326     IL_0001:  ldc.i4.0
327     IL_0002:  stloc.1
328     IL_0003:  ldc.i4 0
329     IL_0008:  stloc.2
330     IL_0009:  ldloc.2
331     IL_000a:  ldsfld int32 BoterKaasEieren.Bord::DIM
332     IL_000f:  clt
333     IL_0011:  brfalse IL_001f
334
335     IL_0016:  ldloc.1
336     IL_0017:  ldc.i4.0
337     IL_0018:  ceq
338     IL_001a:  br IL_0020
339
340     IL_001f:  ldc.i4.0
341     IL_0020:  brfalse IL_006f
342
343     IL_0025:  ldc.i4 0
344     IL_002a:  stloc.3
345     IL_002b:  ldc.i4.1
346     IL_002c:  stloc.1
347     IL_002d:  ldloc.3
348     IL_002e:  ldsfld int32 BoterKaasEieren.Bord::DIM
349     IL_0033:  clt
350     IL_0035:  brfalse IL_0040
351
352     IL_003a:  ldloc.1
353     IL_003b:  br IL_0041
354
355     IL_0040:  ldc.i4.0
356     IL_0041:  brfalse IL_0062
357
358     IL_0046:  ldarg.0
359     IL_0047:  ldloc.3
360     IL_0048:  ldloc.2
361     IL_0049:  call instance class BoterKaasEieren.Mark class
            BoterKaasEieren.Bord::GetVakje(int32, int32)
362     IL_004e:  ldarg 1
363
364     IL_0052:  ceq
365     IL_0054:  stloc.1
366     IL_0055:  ldloc.3
367     IL_0056:  ldc.i4 1
368     IL_005b:  add
```

```
369       IL_005c:  stloc.3
370       IL_005d:  br IL_002d
371
372       IL_0062:  ldloc.2
373       IL_0063:  ldc.i4 1
374       IL_0068:  add
375       IL_0069:  stloc.2
376       IL_006a:  br IL_0009
377
378       IL_006f:  ldloc.1
379       IL_0070:  ret
380       } // end of method Bord::HeeftKolom
381
382       // method line 12
383       .method public hidebysig
384             instance default bool HeeftDiagonaal (class
                  BoterKaasEieren.Mark mark)  cil managed
385       {
386           // Method begins at RVA 0x2368
387       // Code size 141 (0x8d)
388       .maxstack 11
389       .locals init (
390           object  V_0,
391           bool    V_1,
392           bool    V_2,
393           int32   V_3)
394       IL_0000:  nop
395       IL_0001:  ldc.i4 0
396       IL_0006:  stloc.3
397       IL_0007:  ldc.i4.1
398       IL_0008:  stloc.2
399       IL_0009:  ldloc.2
400       IL_000a:  stloc.1
401       IL_000b:  ldloc.3
402       IL_000c:  ldsfld int32 BoterKaasEieren.Bord::DIM
403       IL_0011:  clt
404       IL_0013:  brfalse IL_002a
405
406       IL_0018:  ldloc.1
407       IL_0019:  brtrue IL_0024
408
409       IL_001e:  ldloc.2
410       IL_001f:  br IL_0025
411
412       IL_0024:  ldc.i4.1
413       IL_0025:  br IL_002b
414
415       IL_002a:  ldc.i4.0
416       IL_002b:  brfalse IL_007f
417
418       IL_0030:  ldloc.1
419       IL_0031:  brfalse IL_0049
420
421       IL_0036:  ldarg.0
422       IL_0037:  ldloc.3
423       IL_0038:  ldloc.3
424       IL_0039:  call instance class BoterKaasEieren.Mark class
              BoterKaasEieren.Bord::GetVakje(int32, int32)
425       IL_003e:  ldarg 1
426
427       IL_0042:  ceq
428       IL_0044:  br IL_004a
```

```
429
430       IL_0049:  ldc.i4.0
431       IL_004a:  stloc.1
432       IL_004b:  ldloc.2
433       IL_004c:  brfalse IL_0070
434
435       IL_0051:  ldarg.0
436       IL_0052:  ldloc.3
437       IL_0053:  ldsfld int32 BoterKaasEieren.Bord::DIM
438       IL_0058:  ldloc.3
439       IL_0059:  sub
440       IL_005a:  ldc.i4 1
441       IL_005f:  sub
442       IL_0060:  call instance class BoterKaasEieren.Mark class
              BoterKaasEieren.Bord::GetVakje(int32, int32)
443       IL_0065:  ldarg 1
444
445       IL_0069:  ceq
446       IL_006b:  br IL_0071
447
448       IL_0070:  ldc.i4.0
449       IL_0071:  stloc.2
450       IL_0072:  ldloc.3
451       IL_0073:  ldc.i4 1
452       IL_0078:  add
453       IL_0079:  stloc.3
454       IL_007a:  br IL_000b
455
456       IL_007f:  ldloc.1
457       IL_0080:  brtrue IL_008b
458
459       IL_0085:  ldloc.2
460       IL_0086:  br IL_008c
461
462       IL_008b:  ldc.i4.1
463       IL_008c:  ret
464       } // end of method Bord::HeeftDiagonaal
465
466       // method line 13
467       .method public hidebysig
468             instance default bool HeeftWinnaar ()  cil managed
469       {
470           // Method begins at RVA 0x2404
471       // Code size 35 (0x23)
472       .maxstack 6
473       .locals init (
474           object  V_0)
475       IL_0000:  nop
476       IL_0001:  ldarg.0
477       IL_0002:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::XX
478       IL_0007:  call instance bool class BoterKaasEieren.Bord::
              HeeftGewonnen(class BoterKaasEieren.Mark)
479       IL_000c:  brtrue IL_0021
480
481       IL_0011:  ldarg.0
482       IL_0012:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::OO
483       IL_0017:  call instance bool class BoterKaasEieren.Bord::
              HeeftGewonnen(class BoterKaasEieren.Mark)
484       IL_001c:  br IL_0022
485
```

```
486       IL_0021:  ldc.i4.1
487       IL_0022:  ret
488       } // end of method Bord::HeeftWinnaar
489
490       // method line 14
491       .method public hidebysig
492             instance default bool GameOver ()  cil managed
493       {
494           // Method begins at RVA 0x2434
495       // Code size 25 (0x19)
496       .maxstack 6
497       .locals init (
498           object  V_0)
499       IL_0000:  nop
500       IL_0001:  ldarg.0
501       IL_0002:  call instance bool class BoterKaasEieren.Bord::
              HeeftWinnaar()
502       IL_0007:  brtrue IL_0017
503
504       IL_000c:  ldarg.0
505       IL_000d:  call instance bool class BoterKaasEieren.Bord::IsVol
              ()
506       IL_0012:  br IL_0018
507
508       IL_0017:  ldc.i4.1
509       IL_0018:  ret
510       } // end of method Bord::GameOver
511
512       // method line 15
513       .method public hidebysig
514             instance default bool IsVol ()  cil managed
515       {
516           // Method begins at RVA 0x245c
517       // Code size 73 (0x49)
518       .maxstack 6
519       .locals init (
520           object  V_0,
521           int32   V_1,
522           bool    V_2)
523       IL_0000:  nop
524       IL_0001:  ldc.i4 0
525       IL_0006:  stloc.1
526       IL_0007:  ldc.i4.1
527       IL_0008:  stloc.2
528       IL_0009:  ldloc.1
529       IL_000a:  ldsfld int32 BoterKaasEieren.Bord::DIM
530       IL_000f:  ldsfld int32 BoterKaasEieren.Bord::DIM
531       IL_0014:  mul
532       IL_0015:  clt
533       IL_0017:  brfalse IL_0022
534
535       IL_001c:  ldloc.2
536       IL_001d:  br IL_0023
537
538       IL_0022:  ldc.i4.0
539       IL_0023:  brfalse IL_0047
540
541       IL_0028:  ldarg.0
542       IL_0029:  ldloc.1
543       IL_002a:  call instance class BoterKaasEieren.Mark class
              BoterKaasEieren.Bord::GetVakje(int32)
```

```
544       IL_002f:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::EMPTY
545       IL_0034:  ceq
546       IL_0036:  ldc.i4.0
547       IL_0037:  ceq
548       IL_0039:  stloc.2
549       IL_003a:  ldloc.1
550       IL_003b:  ldc.i4 1
551       IL_0040:  add
552       IL_0041:  stloc.1
553       IL_0042:  br IL_0009
554
555       IL_0047:  ldloc.2
556       IL_0048:  ret
557       } // end of method Bord::IsVol
558
559       // method line 16
560       .method public hidebysig
561             instance default bool HeeftGewonnen (class
                  BoterKaasEieren.Mark m)  cil managed
562       {
563           // Method begins at RVA 0x24b4
564       // Code size 54 (0x36)
565       .maxstack 9
566       .locals init (
567           object  V_0)
568       IL_0000:  nop
569       IL_0001:  ldarg.0
570       IL_0002:  ldarg 1
571
572       IL_0006:  call instance bool class BoterKaasEieren.Bord::
              HeeftRij(class BoterKaasEieren.Mark)
573       IL_000b:  brtrue IL_001f
574
575       IL_0010:  ldarg.0
576       IL_0011:  ldarg 1
577
578       IL_0015:  call instance bool class BoterKaasEieren.Bord::
              HeeftKolom(class BoterKaasEieren.Mark)
579       IL_001a:  br IL_0020
580
581       IL_001f:  ldc.i4.1
582       IL_0020:  brtrue IL_0034
583
584       IL_0025:  ldarg.0
585       IL_0026:  ldarg 1
586
587       IL_002a:  call instance bool class BoterKaasEieren.Bord::
              HeeftDiagonaal(class BoterKaasEieren.Mark)
588       IL_002f:  br IL_0035
589
590       IL_0034:  ldc.i4.1
591       IL_0035:  ret
592       } // end of method Bord::HeeftGewonnen
593
594       // method line 17
595       .method public hidebysig
596             instance default bool IsVakje (int32 rij, int32 kolom)
                  cil managed
597       {
598           // Method begins at RVA 0x24f8
599       // Code size 22 (0x16)
```

```
600     .maxstack 5
601     .locals init (
602         object  V_0)
603     IL_0000:  nop
604     IL_0001:  ldarg.0
605     IL_0002:  ldarg.0
606     IL_0003:  ldarg 1
607
608     IL_0007:  ldarg 2
609
610     IL_000b:  call instance int32 class BoterKaasEieren.Bord::Index
            (int32, int32)
611     IL_0010:  call instance bool class BoterKaasEieren.Bord::
            IsVakje(int32)
612     IL_0015:  ret
613     } // end of method Bord::IsVakje
614
615     // method line 18
616     .method public hidebysig
617           instance default bool IsVakje (int32 i)  cil managed
618     {
619         // Method begins at RVA 0x251c
620     // Code size 44 (0x2c)
621     .maxstack 4
622     .locals init (
623         object  V_0)
624     IL_0000:  nop
625     IL_0001:  ldarg 1
626
627     IL_0005:  ldsfld int32 BoterKaasEieren.Bord::DIM
628     IL_000a:  ldsfld int32 BoterKaasEieren.Bord::DIM
629     IL_000f:  mul
630     IL_0010:  clt
631     IL_0012:  brfalse IL_002a
632
633     IL_0017:  ldarg 1
634
635     IL_001b:  ldc.i4 0
636     IL_0020:  clt
637     IL_0022:  ldc.i4.0
638     IL_0023:  ceq
639     IL_0025:  br IL_002b
640
641     IL_002a:  ldc.i4.0
642     IL_002b:  ret
643     } // end of method Bord::IsVakje
644
645     // method line 19
646     .method public hidebysig
647           instance default bool IsLeegVakje (int32 rij, int32
                kolom)  cil managed
648     {
649         // Method begins at RVA 0x2554
650     // Code size 22 (0x16)
651     .maxstack 5
652     .locals init (
653         object  V_0)
654     IL_0000:  nop
655     IL_0001:  ldarg.0
656     IL_0002:  ldarg.0
657     IL_0003:  ldarg 1
658
```

```
659     IL_0007:  ldarg 2
660
661     IL_000b:  call instance int32 class BoterKaasEieren.Bord::Index
            (int32, int32)
662     IL_0010:  call instance bool class BoterKaasEieren.Bord::
            IsLeegVakje(int32)
663     IL_0015:  ret
664     } // end of method Bord::IsLeegVakje
665
666     // method line 20
667     .method public hidebysig
668           instance default bool IsLeegVakje (int32 i)  cil managed
669     {
670         // Method begins at RVA 0x2578
671     // Code size 24 (0x18)
672     .maxstack 3
673     .locals init (
674         object  V_0)
675     IL_0000:  nop
676     IL_0001:  ldarg.0
677     IL_0002:  ldfld class BoterKaasEieren.Mark[] BoterKaasEieren.
            Bord::bord
678     IL_0007:  ldarg 1
679
680     IL_000b:  ldelem.any BoterKaasEieren.Mark
681     IL_0010:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::EMPTY
682     IL_0015:  ceq
683     IL_0017:  ret
684     } // end of method Bord::IsLeegVakje
685
686     // method line 21
687     .method public hidebysig
688           instance default bool IsCorrecteZet (int32 i)  cil
                managed
689     {
690         // Method begins at RVA 0x259c
691     // Code size 33 (0x21)
692     .maxstack 6
693     .locals init (
694         object  V_0)
695     IL_0000:  nop
696     IL_0001:  ldarg.0
697     IL_0002:  ldarg 1
698
699     IL_0006:  call instance bool class BoterKaasEieren.Bord::
            IsVakje(int32)
700     IL_000b:  brfalse IL_001f
701
702     IL_0010:  ldarg.0
703     IL_0011:  ldarg 1
704
705     IL_0015:  call instance bool class BoterKaasEieren.Bord::
            IsLeegVakje(int32)
706     IL_001a:  br IL_0020
707
708     IL_001f:  ldc.i4.0
709     IL_0020:  ret
710     } // end of method Bord::IsCorrecteZet
711
712     // method line 22
713     .method public hidebysig
```

```
714             instance default bool IsCorrecteZet (int32 rij, int32
                    kol)  cil managed
715     {
716         // Method begins at RVA 0x25cc
717     // Code size 22 (0x16)
718     .maxstack 5
719     .locals init (
720         object  V_0)
721     IL_0000:  nop
722     IL_0001:  ldarg.0
723     IL_0002:  ldarg.0
724     IL_0003:  ldarg 1
725
726     IL_0007:  ldarg 2
727
728     IL_000b:  call instance int32 class BoterKaasEieren.Bord::Index
            (int32, int32)
729     IL_0010:  call instance bool class BoterKaasEieren.Bord::
            IsCorrecteZet(int32)
730     IL_0015:  ret
731     } // end of method Bord::IsCorrecteZet
732
733   } // end of class BoterKaasEieren.Bord
734 }
735
736 .namespace BoterKaasEieren
737 {
738   .class public auto ansi Mark
739     extends [mscorlib]System.Object
740   {
741     .field  public static  initonly  class BoterKaasEieren.Mark XX
742     .field  public static  initonly  class BoterKaasEieren.Mark OO
743     .field  public static  initonly  class BoterKaasEieren.Mark
            EMPTY
744     .field  private  string sign
745
746     // method line 23
747     .method private hidebysig  specialname  rtspecialname
748             instance default void '.ctor' (string sign)  cil managed
749     {
750         // Method begins at RVA 0x25f0
751     // Code size 17 (0x11)
752     .maxstack 4
753     .locals init (
754         object  V_0)
755     IL_0000:  ldarg.0
756     IL_0001:  call instance void object::'.ctor'()
757     IL_0006:  ldarg.0
758     IL_0007:  ldarg 1
759
760     IL_000b:  stfld string BoterKaasEieren.Mark::sign
761     IL_0010:  ret
762     } // end of method Mark::.ctor
763
764     // method line 24
765     .method public static  specialname  rtspecialname
766             default void '.cctor' ()  cil managed
767     {
768         // Method begins at RVA 0x2610
769     // Code size 47 (0x2f)
770     .maxstack 2
771     .locals init (
```

```
772         object  V_0)
773     IL_0000:  nop
774     IL_0001:  ldstr "X"
775     IL_0006:  newobj instance void class BoterKaasEieren.Mark::'.
            ctor'(string)
776     IL_000b:  stsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::XX
777     IL_0010:  ldstr "O"
778     IL_0015:  newobj instance void class BoterKaasEieren.Mark::'.
            ctor'(string)
779     IL_001a:  stsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::OO
780     IL_001f:  ldstr " "
781     IL_0024:  newobj instance void class BoterKaasEieren.Mark::'.
            ctor'(string)
782     IL_0029:  stsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::EMPTY
783     IL_002e:  ret
784     } // end of method Mark::.cctor
785
786     // method line 25
787     .method public hidebysig
788           instance default class BoterKaasEieren.Mark Other ()
                cil managed
789     {
790         // Method begins at RVA 0x264c
791     // Code size 30 (0x1e)
792     .maxstack 3
793     .locals init (
794         object  V_0)
795     IL_0000:  nop
796     IL_0001:  ldarg.0
797     IL_0002:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::XX
798     IL_0007:  ceq
799     IL_0009:  brfalse IL_0018
800
801     IL_000e:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::OO
802     IL_0013:  br IL_001d
803
804     IL_0018:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
            Mark::XX
805     IL_001d:  ret
806     } // end of method Mark::Other
807
808     // method line 26
809     .method public virtual  hidebysig
810           instance default string ToString ()  cil managed
811     {
812         // Method begins at RVA 0x2678
813     // Code size 8 (0x8)
814     .maxstack 2
815     .locals init (
816         object  V_0)
817     IL_0000:  nop
818     IL_0001:  ldarg.0
819     IL_0002:  ldfld string BoterKaasEieren.Mark::sign
820     IL_0007:  ret
821     } // end of method Mark::ToString
822
823 } // end of class BoterKaasEieren.Mark
```

```
824 }
825
826 .namespace BoterKaasEieren
827 {
828   .class public auto ansi MensSpeler
829     extends BoterKaasEieren.Speler
830   {
831     .field  private  int32 tempMove
832
833     // method line 27
834     .method public hidebysig  specialname  rtspecialname
835            instance default void '.ctor' (string naam, class
                BoterKaasEieren.Mark mark)  cil managed
836     {
837         // Method begins at RVA 0x268c
838     // Code size 26 (0x1a)
839     .maxstack 4
840     .locals init (
841         object  V_0)
842     IL_0000:  ldarg.0
843     IL_0001:  ldarg 1
844
845     IL_0005:  ldarg 2
846
847     IL_0009:  call instance void class BoterKaasEieren.Speler::'.
          ctor'(string, class BoterKaasEieren.Mark)
848     IL_000e:  ldarg.0
849     IL_000f:  ldc.i4 -1
850     IL_0014:  stfld int32 BoterKaasEieren.MensSpeler::tempMove
851     IL_0019:  ret
852     } // end of method MensSpeler::.ctor
853
854     // method line 28
855     .method public virtual  hidebysig
856            instance default int32 BepaalZet (class BoterKaasEieren.
                Bord bord)  cil managed
857     {
858         // Method begins at RVA 0x26b4
859     // Code size 68 (0x44)
860     .maxstack 7
861     .locals init (
862         object  V_0,
863         int32   V_1)
864     IL_0000:  nop
865     IL_0001:  ldarg.0
866     IL_0002:  call void class [mscorlib]System.Threading.Monitor::
          Enter(object)
867     IL_0007:  ldarg 1
868
869     IL_000b:  ldarg.0
870     IL_000c:  ldfld int32 BoterKaasEieren.MensSpeler::tempMove
871     IL_0011:  call instance bool class BoterKaasEieren.Bord::
          IsCorrecteZet(int32)
872     IL_0016:  ldc.i4.0
873     IL_0017:  ceq
874     IL_0019:  brfalse IL_002a
875
876     IL_001e:  ldarg.0
877     IL_001f:  call bool class [mscorlib]System.Threading.Monitor::
          Wait(object)
878     IL_0024:  pop
879     IL_0025:  br IL_0007
```

```
880
881       IL_002a:  ldarg.0
882       IL_002b:  call void class [mscorlib]System.Threading.Monitor::
              Exit(object)
883       IL_0030:  ldarg.0
884       IL_0031:  ldfld int32 BoterKaasEieren.MensSpeler::tempMove
885       IL_0036:  stloc.1
886       IL_0037:  ldarg.0
887       IL_0038:  ldc.i4 -1
888       IL_003d:  stfld int32 BoterKaasEieren.MensSpeler::tempMove
889       IL_0042:  ldloc.1
890       IL_0043:  ret
891       } // end of method MensSpeler::BepaalZet
892
893       // method line 29
894       .method public hidebysig
895             instance default int32 GetTempMove ()  cil managed
896       {
897           // Method begins at RVA 0x2704
898       // Code size 8 (0x8)
899       .maxstack 2
900       .locals init (
901           object  V_0)
902       IL_0000:  nop
903       IL_0001:  ldarg.0
904       IL_0002:  ldfld int32 BoterKaasEieren.MensSpeler::tempMove
905       IL_0007:  ret
906       } // end of method MensSpeler::GetTempMove
907
908       // method line 30
909       .method public hidebysig
910             instance default void SetTempMove (int32 tempMove)  cil
                  managed
911       {
912           // Method begins at RVA 0x2718
913       // Code size 12 (0xc)
914       .maxstack 2
915       .locals init (
916           object  V_0)
917       IL_0000:  nop
918       IL_0001:  ldarg.0
919       IL_0002:  ldarg 1
920
921       IL_0006:  stfld int32 BoterKaasEieren.MensSpeler::tempMove
922       IL_000b:  ret
923       } // end of method MensSpeler::SetTempMove
924
925     } // end of class BoterKaasEieren.MensSpeler
926   }
927
928   .namespace BoterKaasEieren
929   {
930     .class public auto ansi Spel
931       extends [mscorlib]System.Object
932     {
933       .field  public static  initonly  int32 AANTAL_SPELERS
934       .field  private  class BoterKaasEieren.Speler[] spelers
935       .field  private  int32 huidig
936       .field  private  class BoterKaasEieren.Bord bord
937       .field  private  class [mscorlib]System.Collections.ArrayList
              listeners
938
```

```
939     // method line 31
940     .method public hidebysig  specialname  rtspecialname
941             instance default void '.ctor' (class BoterKaasEieren.
                    Speler s1, class BoterKaasEieren.Speler s2)  cil
                    managed
942     {
943         // Method begins at RVA 0x2730
944     // Code size 75 (0x4b)
945     .maxstack 7
946     .locals init (
947         object  V_0)
948     IL_0000:  ldarg.0
949     IL_0001:  call instance void object::'.ctor'()
950     IL_0006:  ldarg.0
951     IL_0007:  ldc.i4 2
952     IL_000c:  newarr BoterKaasEieren.Speler
953     IL_0011:  dup
954     IL_0012:  ldc.i4 0
955     IL_0017:  ldarg 1
956
957     IL_001b:  stelem.any BoterKaasEieren.Speler
958     IL_0020:  dup
959     IL_0021:  ldc.i4 1
960     IL_0026:  ldarg 2
961
962     IL_002a:  stelem.any BoterKaasEieren.Speler
963     IL_002f:  stfld class BoterKaasEieren.Speler[] BoterKaasEieren.
            Spel::spelers
964     IL_0034:  ldarg.0
965     IL_0035:  newobj instance void class [mscorlib]System.
            Collections.ArrayList::'.ctor'()
966     IL_003a:  stfld class [mscorlib]System.Collections.ArrayList
            BoterKaasEieren.Spel::listeners
967     IL_003f:  ldarg.0
968     IL_0040:  newobj instance void class BoterKaasEieren.Bord::'.
            ctor'()
969     IL_0045:  stfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
            ::bord
970     IL_004a:  ret
971     } // end of method Spel::.ctor
972
973     // method line 32
974     .method public static  specialname  rtspecialname
975             default void '.cctor' ()  cil managed
976     {
977         // Method begins at RVA 0x2788
978     // Code size 12 (0xc)
979     .maxstack 1
980     .locals init (
981         object  V_0)
982     IL_0000:  nop
983     IL_0001:  ldc.i4 2
984     IL_0006:  stsfld int32 BoterKaasEieren.Spel::AANTAL_SPELERS
985     IL_000b:  ret
986     } // end of method Spel::.cctor
987
988     // method line 33
989     .method public hidebysig
990             instance default class BoterKaasEieren.Bord GetBord ()
                    cil managed
991     {
992         // Method begins at RVA 0x27a0
```

```
993       // Code size 8 (0x8)
994       .maxstack 2
995       .locals init (
996           object  V_0)
997       IL_0000:  nop
998       IL_0001:  ldarg.0
999       IL_0002:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1000      IL_0007:  ret
1001      } // end of method Spel::GetBord
1002
1003      // method line 34
1004      .method public hidebysig
1005            instance default int32 GetHuidig ()  cil managed
1006      {
1007          // Method begins at RVA 0x27b4
1008      // Code size 8 (0x8)
1009      .maxstack 2
1010      .locals init (
1011          object  V_0)
1012      IL_0000:  nop
1013      IL_0001:  ldarg.0
1014      IL_0002:  ldfld int32 BoterKaasEieren.Spel::huidig
1015      IL_0007:  ret
1016      } // end of method Spel::GetHuidig
1017
1018      // method line 35
1019      .method public hidebysig
1020            instance default class BoterKaasEieren.Speler
                  GetHuidigeSpeler ()  cil managed
1021      {
1022          // Method begins at RVA 0x27c8
1023      // Code size 19 (0x13)
1024      .maxstack 3
1025      .locals init (
1026          object  V_0)
1027      IL_0000:  nop
1028      IL_0001:  ldarg.0
1029      IL_0002:  ldfld class BoterKaasEieren.Speler[] BoterKaasEieren.
              Spel::spelers
1030      IL_0007:  ldarg.0
1031      IL_0008:  ldfld int32 BoterKaasEieren.Spel::huidig
1032      IL_000d:  ldelem.any BoterKaasEieren.Speler
1033      IL_0012:  ret
1034      } // end of method Spel::GetHuidigeSpeler
1035
1036      // method line 36
1037      .method public hidebysig
1038            instance default class BoterKaasEieren.Speler[]
                  GetSpelers ()  cil managed
1039      {
1040          // Method begins at RVA 0x27e8
1041      // Code size 8 (0x8)
1042      .maxstack 2
1043      .locals init (
1044          object  V_0)
1045      IL_0000:  nop
1046      IL_0001:  ldarg.0
1047      IL_0002:  ldfld class BoterKaasEieren.Speler[] BoterKaasEieren.
              Spel::spelers
1048      IL_0007:  ret
1049      } // end of method Spel::GetSpelers
```

```
1050
1051        // method line 37
1052        .method public hidebysig
1053                instance default void Reset ()  cil managed
1054        {
1055            // Method begins at RVA 0x27fc
1056        // Code size 24 (0x18)
1057        .maxstack 4
1058        .locals init (
1059            object  V_0)
1060        IL_0000:  nop
1061        IL_0001:  ldarg.0
1062        IL_0002:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1063        IL_0007:  call instance void class BoterKaasEieren.Bord::Reset
              ()
1064        IL_000c:  ldarg.0
1065        IL_000d:  ldc.i4 0
1066        IL_0012:  stfld int32 BoterKaasEieren.Spel::huidig
1067        IL_0017:  ret
1068        } // end of method Spel::Reset
1069
1070        // method line 38
1071        .method public hidebysig
1072                instance default void Play ()  cil managed
1073        {
1074            // Method begins at RVA 0x2820
1075        // Code size 157 (0x9d)
1076        .maxstack 14
1077        .locals init (
1078            object  V_0)
1079        IL_0000:  nop
1080        IL_0001:  ldarg.0
1081        IL_0002:  ldnull
1082        IL_0003:  call instance void class BoterKaasEieren.Spel::Update
              (object)
1083        IL_0008:  ldarg.0
1084        IL_0009:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1085        IL_000e:  call instance bool class BoterKaasEieren.Bord::
              GameOver()
1086        IL_0013:  ldc.i4.0
1087        IL_0014:  ceq
1088        IL_0016:  brfalse IL_0060
1089
1090        IL_001b:  ldarg.0
1091        IL_001c:  ldfld class BoterKaasEieren.Speler[] BoterKaasEieren.
              Spel::spelers
1092        IL_0021:  ldarg.0
1093        IL_0022:  ldfld int32 BoterKaasEieren.Spel::huidig
1094        IL_0027:  ldelem.any BoterKaasEieren.Speler
1095        IL_002c:  ldarg.0
1096        IL_002d:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1097        IL_0032:  call instance void class BoterKaasEieren.Speler::
              DoeZet(class BoterKaasEieren.Bord)
1098        IL_0037:  ldarg.0
1099        IL_0038:  ldarg.0
1100        IL_0039:  ldfld int32 BoterKaasEieren.Spel::huidig
1101        IL_003e:  ldc.i4 1
1102        IL_0043:  add
1103        IL_0044:  ldsfld int32 BoterKaasEieren.Spel::AANTAL_SPELERS
```

```
1104      IL_0049:  rem
1105      IL_004a:  stfld int32 BoterKaasEieren.Spel::huidig
1106      IL_004f:  ldarg.0
1107      IL_0050:  ldarg.0
1108      IL_0051:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1109      IL_0056:  call instance void class BoterKaasEieren.Spel::Update
              (object)
1110      IL_005b:  br IL_0008
1111
1112      IL_0060:  ldarg.0
1113      IL_0061:  ldarg.0
1114      IL_0062:  ldfld class BoterKaasEieren.Bord BoterKaasEieren.Spel
              ::bord
1115      IL_0067:  call instance bool class BoterKaasEieren.Bord::
              HeeftWinnaar()
1116      IL_006c:  brfalse IL_008d
1117
1118      IL_0071:  ldarg.0
1119      IL_0072:  ldfld int32 BoterKaasEieren.Spel::huidig
1120      IL_0077:  ldc.i4 1
1121      IL_007c:  add
1122      IL_007d:  ldsfld int32 BoterKaasEieren.Spel::AANTAL_SPELERS
1123      IL_0082:  rem
1124      IL_0083:  box [mscorlib]System.Int32
1125      IL_0088:  br IL_0097
1126
1127      IL_008d:  ldc.i4 -1
1128      IL_0092:  box [mscorlib]System.Int32
1129      IL_0097:  call instance void class BoterKaasEieren.Spel::Update
              (object)
1130      IL_009c:  ret
1131      } // end of method Spel::Play
1132
1133      // method line 39
1134      .method public hidebysig
1135            instance default void AddObserver (class BoterKaasEieren
                  .IMessageListener l)  cil managed
1136      {
1137          // Method begins at RVA 0x28cc
1138      // Code size 18 (0x12)
1139      .maxstack 3
1140      .locals init (
1141          object  V_0)
1142      IL_0000:  nop
1143      IL_0001:  ldarg.0
1144      IL_0002:  ldfld class [mscorlib]System.Collections.ArrayList
              BoterKaasEieren.Spel::listeners
1145      IL_0007:  ldarg 1
1146
1147      IL_000b:  callvirt instance int32 class [mscorlib]System.
              Collections.ArrayList::Add(object)
1148      IL_0010:  pop
1149      IL_0011:  ret
1150      } // end of method Spel::AddObserver
1151
1152      // method line 40
1153      .method public hidebysig
1154            instance default void RemoveObserver (class
                  BoterKaasEieren.IMessageListener l)  cil managed
1155      {
1156          // Method begins at RVA 0x28ec
```

```
1157      // Code size 17 (0x11)
1158      .maxstack 3
1159      .locals init (
1160          object  V_0)
1161      IL_0000:  nop
1162      IL_0001:  ldarg.0
1163      IL_0002:  ldfld class [mscorlib]System.Collections.ArrayList
              BoterKaasEieren.Spel::listeners
1164      IL_0007:  ldarg 1
1165
1166      IL_000b:  callvirt instance void class [mscorlib]System.
              Collections.ArrayList::Remove(object)
1167      IL_0010:  ret
1168      } // end of method Spel::RemoveObserver
1169
1170      // method line 41
1171      .method public hidebysig
1172            instance default void Update (object msg)  cil managed
1173      {
1174          // Method begins at RVA 0x290c
1175      // Code size 113 (0x71)
1176      .maxstack 15
1177      .locals init (
1178          object  V_0,
1179          class BoterKaasEieren.IMessageListener  V_1)
1180      IL_0000:  nop
1181      IL_0001:  ldarg.0
1182      IL_0002:  ldfld class [mscorlib]System.Collections.ArrayList
              BoterKaasEieren.Spel::listeners
1183      IL_0007:  callvirt instance class [mscorlib]System.Collections.
              IEnumerator class [mscorlib]System.Collections.IEnumerable
              ::GetEnumerator()
1184      IL_000c:  dup
1185      IL_000d:  callvirt instance bool class [mscorlib]System.
              Collections.IEnumerator::MoveNext()
1186      IL_0012:  brfalse IL_006f
1187
1188      IL_0017:  dup
1189      IL_0018:  callvirt instance object class [mscorlib]System.
              Collections.IEnumerator::get_Current()
1190      IL_001d:  castclass BoterKaasEieren.IMessageListener
1191      IL_0022:  stloc.1
1192      IL_0023:  ldloc.1
1193      IL_0024:  callvirt instance bool class [System]System.
              ComponentModel.ISynchronizeInvoke::get_InvokeRequired()
1194      IL_0029:  brfalse IL_0060
1195
1196      IL_002e:  ldloc.1
1197      IL_002f:  ldloc.1
1198      IL_0030:  dup
1199      IL_0031:  ldvirtftn instance void class BoterKaasEieren.
              IMessageListener::MessageReceived(object)
1200      IL_0037:  newobj instance void class [mscorlib]System.Threading
              .ParameterizedThreadStart::'.ctor'(object, native int)
1201      IL_003c:  ldc.i4 1
1202      IL_0041:  newarr [mscorlib]System.Object
1203      IL_0046:  dup
1204      IL_0047:  ldc.i4 0
1205      IL_004c:  ldarg 1
1206
1207      IL_0050:  stelem.any [mscorlib]System.Object
```

```
1208      IL_0055:  callvirt instance object class [System]System.
              ComponentModel.ISynchronizeInvoke::Invoke(class [mscorlib]
              System.Delegate, object[])
1209      IL_005a:  pop
1210      IL_005b:  br IL_006a
1211
1212      IL_0060:  ldloc.1
1213      IL_0061:  ldarg 1
1214
1215      IL_0065:  callvirt instance void class BoterKaasEieren.
              IMessageListener::MessageReceived(object)
1216      IL_006a:  br IL_000c
1217
1218      IL_006f:  pop
1219      IL_0070:  ret
1220      } // end of method Spel::Update
1221
1222    } // end of class BoterKaasEieren.Spel
1223  }
1224
1225  .namespace BoterKaasEieren
1226  {
1227    .class public auto ansi ComputerSpeler
1228      extends BoterKaasEieren.Speler
1229    {
1230      .field  private  class BoterKaasEieren.IStrategie strategie
1231
1232      // method line 42
1233      .method public hidebysig  specialname  rtspecialname
1234              instance default void '.ctor' (string naam, class
                    BoterKaasEieren.Mark mark, class BoterKaasEieren.
                    IStrategie strategie)  cil managed
1235      {
1236          // Method begins at RVA 0x298c
1237      // Code size 25 (0x19)
1238      .maxstack 4
1239      .locals init (
1240          object  V_0)
1241      IL_0000:  ldarg.0
1242      IL_0001:  ldarg 1
1243
1244      IL_0005:  ldarg 2
1245
1246      IL_0009:  call instance void class BoterKaasEieren.Speler::'.
              ctor'(string, class BoterKaasEieren.Mark)
1247      IL_000e:  ldarg.0
1248      IL_000f:  ldarg 3
1249
1250      IL_0013:  stfld class BoterKaasEieren.IStrategie
              BoterKaasEieren.ComputerSpeler::strategie
1251      IL_0018:  ret
1252      } // end of method ComputerSpeler::.ctor
1253
1254      // method line 43
1255      .method public virtual  hidebysig
1256              instance default int32 BepaalZet (class BoterKaasEieren.
                    Bord bord)  cil managed
1257      {
1258          // Method begins at RVA 0x29b4
1259      // Code size 23 (0x17)
1260      .maxstack 6
1261      .locals init (
```

```
1262          object  V_0)
1263      IL_0000:  nop
1264      IL_0001:  ldarg.0
1265      IL_0002:  ldfld class BoterKaasEieren.IStrategie
              BoterKaasEieren.ComputerSpeler::strategie
1266      IL_0007:  ldarg 1
1267
1268      IL_000b:  ldarg.0
1269      IL_000c:  call instance class BoterKaasEieren.Mark class
              BoterKaasEieren.Speler::GetMark()
1270      IL_0011:  callvirt instance int32 class BoterKaasEieren.
              IStrategie::BerekenZet(class BoterKaasEieren.Bord, class
              BoterKaasEieren.Mark)
1271      IL_0016:  ret
1272      } // end of method ComputerSpeler::BepaalZet
1273
1274      // method line 44
1275      .method public hidebysig
1276            instance default class BoterKaasEieren.IStrategie
                  GetStrategie ()  cil managed
1277      {
1278          // Method begins at RVA 0x29d8
1279      // Code size 8 (0x8)
1280      .maxstack 2
1281      .locals init (
1282          object  V_0)
1283      IL_0000:  nop
1284      IL_0001:  ldarg.0
1285      IL_0002:  ldfld class BoterKaasEieren.IStrategie
              BoterKaasEieren.ComputerSpeler::strategie
1286      IL_0007:  ret
1287      } // end of method ComputerSpeler::GetStrategie
1288
1289      // method line 45
1290      .method public hidebysig
1291            instance default void SetStrategie (class
                  BoterKaasEieren.IStrategie strategie)  cil managed
1292      {
1293          // Method begins at RVA 0x29ec
1294      // Code size 12 (0xc)
1295      .maxstack 2
1296      .locals init (
1297          object  V_0)
1298      IL_0000:  nop
1299      IL_0001:  ldarg.0
1300      IL_0002:  ldarg 1
1301
1302      IL_0006:  stfld class BoterKaasEieren.IStrategie
              BoterKaasEieren.ComputerSpeler::strategie
1303      IL_000b:  ret
1304      } // end of method ComputerSpeler::SetStrategie
1305
1306    } // end of class BoterKaasEieren.ComputerSpeler
1307  }
1308
1309  .namespace BoterKaasEieren
1310  {
1311    .class interface public auto ansi abstract IStrategie
1312    {
1313
1314      // method line 46
1315      .method public virtual  hidebysig  newslot  abstract
```

```
1316            instance default int32 BerekenZet (class BoterKaasEieren
                    .Bord bord, class BoterKaasEieren.Mark mark)  cil
                    managed
1317    {
1318        // Method begins at RVA 0x0
1319    } // end of method IStrategie::BerekenZet
1320
1321  } // end of class BoterKaasEieren.IStrategie
1322 }
1323
1324 .namespace BoterKaasEieren
1325 {
1326   .class public auto ansi abstract Speler
1327     extends [mscorlib]System.Object
1328   {
1329     .field  private  string naam
1330     .field  private  class BoterKaasEieren.Mark mark
1331
1332     // method line 47
1333     .method family hidebysig  specialname  rtspecialname
1334            instance default void '.ctor' (string naam, class
                    BoterKaasEieren.Mark mark)  cil managed
1335     {
1336         // Method begins at RVA 0x2a04
1337     // Code size 27 (0x1b)
1338     .maxstack 4
1339     .locals init (
1340         object  V_0)
1341     IL_0000:  ldarg.0
1342     IL_0001:  call instance void object::'.ctor'()
1343     IL_0006:  ldarg.0
1344     IL_0007:  ldarg 1
1345
1346     IL_000b:  stfld string BoterKaasEieren.Speler::naam
1347     IL_0010:  ldarg.0
1348     IL_0011:  ldarg 2
1349
1350     IL_0015:  stfld class BoterKaasEieren.Mark BoterKaasEieren.
            Speler::mark
1351     IL_001a:  ret
1352     } // end of method Speler::.ctor
1353
1354     // method line 48
1355     .method public hidebysig
1356            instance default void DoeZet (class BoterKaasEieren.Bord
                    bord)  cil managed
1357     {
1358         // Method begins at RVA 0x2a2c
1359     // Code size 29 (0x1d)
1360     .maxstack 6
1361     .locals init (
1362         object  V_0,
1363         int32   V_1)
1364     IL_0000:  nop
1365     IL_0001:  ldarg.0
1366     IL_0002:  ldarg 1
1367
1368     IL_0006:  callvirt instance int32 class BoterKaasEieren.Speler
            ::BepaalZet(class BoterKaasEieren.Bord)
1369     IL_000b:  stloc.1
1370     IL_000c:  ldarg 1
1371
```

```
1372      IL_0010:  ldloc.1
1373      IL_0011:  ldarg.0
1374      IL_0012:  ldfld class BoterKaasEieren.Mark BoterKaasEieren.
              Speler::mark
1375      IL_0017:  call instance void class BoterKaasEieren.Bord::
              SetVakje(int32, class BoterKaasEieren.Mark)
1376      IL_001c:  ret
1377      } // end of method Speler::DoeZet
1378
1379      // method line 49
1380      .method public hidebysig
1381            instance default string GetNaam ()  cil managed
1382      {
1383          // Method begins at RVA 0x2a58
1384      // Code size 8 (0x8)
1385      .maxstack 2
1386      .locals init (
1387          object  V_0)
1388      IL_0000:  nop
1389      IL_0001:  ldarg.0
1390      IL_0002:  ldfld string BoterKaasEieren.Speler::naam
1391      IL_0007:  ret
1392      } // end of method Speler::GetNaam
1393
1394      // method line 50
1395      .method public hidebysig
1396            instance default class BoterKaasEieren.Mark GetMark ()
                  cil managed
1397      {
1398          // Method begins at RVA 0x2a6c
1399      // Code size 8 (0x8)
1400      .maxstack 2
1401      .locals init (
1402          object  V_0)
1403      IL_0000:  nop
1404      IL_0001:  ldarg.0
1405      IL_0002:  ldfld class BoterKaasEieren.Mark BoterKaasEieren.
              Speler::mark
1406      IL_0007:  ret
1407      } // end of method Speler::GetMark
1408
1409      // method line 51
1410      .method public virtual  hidebysig  abstract
1411            instance default int32 BepaalZet (class BoterKaasEieren.
                  Bord bord)  cil managed
1412      {
1413          // Method begins at RVA 0x0
1414      } // end of method Speler::BepaalZet
1415
1416    } // end of class BoterKaasEieren.Speler
1417  }
1418
1419  .namespace BoterKaasEieren
1420  {
1421    .class public auto ansi BkeGui
1422      extends [System.Windows.Forms]System.Windows.Forms.Form
1423      implements BoterKaasEieren.IMessageListener  {
1424      .field  private  class BoterKaasEieren.Spel spel
1425      .field  private  class [System.Windows.Forms]System.Windows.
              Forms.Button button
1426      .field  private  class [System.Windows.Forms]System.Windows.
              Forms.Button[] tiles
```

```
1427      .field private class [System.Windows.Forms]System.Windows.
              Forms.Label label
1428      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s1radio1
1429      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s1radio2
1430      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s1radio3
1431      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s2radio1
1432      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s2radio2
1433      .field private class [System.Windows.Forms]System.Windows.
              Forms.RadioButton s2radio3
1434
1435      // method line 52
1436      .method public hidebysig specialname rtspecialname
1437              instance default void '.ctor' () cil managed
1438      {
1439          // Method begins at RVA 0x2a80
1440      // Code size 1085 (0x43d)
1441      .maxstack 133
1442      .locals init (
1443          object V_0,
1444          class [System.Windows.Forms]System.Windows.Forms.Panel V_1
                  ,
1445          class [System.Windows.Forms]System.Windows.Forms.Label V_2
                  ,
1446          int32 V_3,
1447          class [System.Windows.Forms]System.Windows.Forms.Button V_4
                  )
1448      IL_0000: ldarg.0
1449      IL_0001: call instance void class [System.Windows.Forms]System
              .Windows.Forms.Form::'.ctor'()
1450      IL_0006: ldarg.0
1451      IL_0007: ldstr "Boter Kaas Eieren"
1452      IL_000c: callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Form::set_Text(string)
1453      IL_0011: ldarg.0
1454      IL_0012: ldc.i4 350
1455      IL_0017: ldc.i4 300
1456      IL_001c: newobj instance void valuetype [System.Drawing]System
              .Drawing.Size::'.ctor'(int32, int32)
1457      IL_0021: call instance void class [System.Windows.Forms]System
              .Windows.Forms.Form::set_Size(valuetype [System.Drawing]
              System.Drawing.Size)
1458      IL_0026: newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.Panel::'.ctor'()
1459      IL_002b: stloc.1
1460      IL_002c: ldloc.1
1461      IL_002d: ldc.i4 80
1462      IL_0032: ldc.i4 80
1463      IL_0037: newobj instance void valuetype [System.Drawing]System
              .Drawing.Size::'.ctor'(int32, int32)
1464      IL_003c: call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Size(valuetype [System.Drawing]
              System.Drawing.Size)
1465      IL_0041: ldloc.1
1466      IL_0042: ldc.i4 10
1467      IL_0047: ldc.i4 10
1468      IL_004c: newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
```

```
1469      IL_0051:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1470      IL_0056:  ldarg.0
1471      IL_0057:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1472      IL_005c:  ldloc.1
1473      IL_005d:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1474      IL_0062:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.Label::'.ctor'()
1475      IL_0067:  stloc.2
1476      IL_0068:  ldloc.2
1477      IL_0069:  ldstr "Speler 1"
1478      IL_006e:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Label::set_Text(string)
1479      IL_0073:  ldloc.1
1480      IL_0074:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1481      IL_0079:  ldloc.2
1482      IL_007a:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1483      IL_007f:  ldarg.0
1484      IL_0080:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.RadioButton::'.ctor'()
1485      IL_0085:  stfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1486      IL_008a:  ldarg.0
1487      IL_008b:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1488      IL_0090:  ldstr "Human"
1489      IL_0095:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.ButtonBase::set_Text(string)
1490      IL_009a:  ldarg.0
1491      IL_009b:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1492      IL_00a0:  ldc.i4.1
1493      IL_00a1:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.RadioButton::set_Checked(bool)
1494      IL_00a6:  ldarg.0
1495      IL_00a7:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1496      IL_00ac:  ldc.i4 0
1497      IL_00b1:  ldc.i4 20
1498      IL_00b6:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
1499      IL_00bb:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1500      IL_00c0:  ldloc.1
1501      IL_00c1:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1502      IL_00c6:  ldarg.0
1503      IL_00c7:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1504      IL_00cc:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
```

```
        System.Windows.Forms]System.Windows.Forms.Control)
1505    IL_00d1:  ldarg.0
1506    IL_00d2:  newobj instance void class [System.Windows.Forms]
        System.Windows.Forms.RadioButton::'.ctor'()
1507    IL_00d7:  stfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio2
1508    IL_00dc:  ldarg.0
1509    IL_00dd:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio2
1510    IL_00e2:  ldstr "Domme AI"
1511    IL_00e7:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.ButtonBase::set_Text(string)
1512    IL_00ec:  ldarg.0
1513    IL_00ed:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio2
1514    IL_00f2:  ldc.i4 0
1515    IL_00f7:  ldc.i4 40
1516    IL_00fc:  newobj instance void valuetype [System.Drawing]System
        .Drawing.Point::'.ctor'(int32, int32)
1517    IL_0101:  call instance void class [System.Windows.Forms]System
        .Windows.Forms.Control::set_Location(valuetype [System.
        Drawing]System.Drawing.Point)
1518    IL_0106:  ldloc.1
1519    IL_0107:  call instance class [System.Windows.Forms]System.
        Windows.Forms.Control/ControlCollection class [System.
        Windows.Forms]System.Windows.Forms.Control::get_Controls()
1520    IL_010c:  ldarg.0
1521    IL_010d:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio2
1522    IL_0112:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.Control/ControlCollection::Add(class [
        System.Windows.Forms]System.Windows.Forms.Control)
1523    IL_0117:  ldarg.0
1524    IL_0118:  newobj instance void class [System.Windows.Forms]
        System.Windows.Forms.RadioButton::'.ctor'()
1525    IL_011d:  stfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio3
1526    IL_0122:  ldarg.0
1527    IL_0123:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio3
1528    IL_0128:  ldstr "Slimme AI"
1529    IL_012d:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.ButtonBase::set_Text(string)
1530    IL_0132:  ldarg.0
1531    IL_0133:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio3
1532    IL_0138:  ldc.i4 0
1533    IL_013d:  ldc.i4 60
1534    IL_0142:  newobj instance void valuetype [System.Drawing]System
        .Drawing.Point::'.ctor'(int32, int32)
1535    IL_0147:  call instance void class [System.Windows.Forms]System
        .Windows.Forms.Control::set_Location(valuetype [System.
        Drawing]System.Drawing.Point)
1536    IL_014c:  ldloc.1
1537    IL_014d:  call instance class [System.Windows.Forms]System.
        Windows.Forms.Control/ControlCollection class [System.
        Windows.Forms]System.Windows.Forms.Control::get_Controls()
1538    IL_0152:  ldarg.0
1539    IL_0153:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s1radio3
1540    IL_0158:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.Control/ControlCollection::Add(class [
```

```
        System.Windows.Forms]System.Windows.Forms.Control)
1541    IL_015d:  newobj instance void class [System.Windows.Forms]
        System.Windows.Forms.Panel::'.ctor'()
1542    IL_0162:  stloc.1
1543    IL_0163:  ldloc.1
1544    IL_0164:  ldc.i4 80
1545    IL_0169:  ldc.i4 80
1546    IL_016e:  newobj instance void valuetype [System.Drawing]System
        .Drawing.Size::'.ctor'(int32, int32)
1547    IL_0173:  call instance void class [System.Windows.Forms]System
        .Windows.Forms.Control::set_Size(valuetype [System.Drawing]
        System.Drawing.Size)
1548    IL_0178:  ldloc.1
1549    IL_0179:  ldc.i4 100
1550    IL_017e:  ldc.i4 10
1551    IL_0183:  newobj instance void valuetype [System.Drawing]System
        .Drawing.Point::'.ctor'(int32, int32)
1552    IL_0188:  call instance void class [System.Windows.Forms]System
        .Windows.Forms.Control::set_Location(valuetype [System.
        Drawing]System.Drawing.Point)
1553    IL_018d:  ldarg.0
1554    IL_018e:  call instance class [System.Windows.Forms]System.
        Windows.Forms.Control/ControlCollection class [System.
        Windows.Forms]System.Windows.Forms.Control::get_Controls()
1555    IL_0193:  ldloc.1
1556    IL_0194:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.Control/ControlCollection::Add(class [
        System.Windows.Forms]System.Windows.Forms.Control)
1557    IL_0199:  newobj instance void class [System.Windows.Forms]
        System.Windows.Forms.Label::'.ctor'()
1558    IL_019e:  stloc.2
1559    IL_019f:  ldloc.2
1560    IL_01a0:  ldstr "Speler 2"
1561    IL_01a5:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.Label::set_Text(string)
1562    IL_01aa:  ldloc.1
1563    IL_01ab:  call instance class [System.Windows.Forms]System.
        Windows.Forms.Control/ControlCollection class [System.
        Windows.Forms]System.Windows.Forms.Control::get_Controls()
1564    IL_01b0:  ldloc.2
1565    IL_01b1:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.Control/ControlCollection::Add(class [
        System.Windows.Forms]System.Windows.Forms.Control)
1566    IL_01b6:  ldarg.0
1567    IL_01b7:  newobj instance void class [System.Windows.Forms]
        System.Windows.Forms.RadioButton::'.ctor'()
1568    IL_01bc:  stfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1569    IL_01c1:  ldarg.0
1570    IL_01c2:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1571    IL_01c7:  ldstr "Human"
1572    IL_01cc:  callvirt instance void class [System.Windows.Forms]
        System.Windows.Forms.ButtonBase::set_Text(string)
1573    IL_01d1:  ldarg.0
1574    IL_01d2:  ldfld class [System.Windows.Forms]System.Windows.
        Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1575    IL_01d7:  ldc.i4.1
1576    IL_01d8:  call instance void class [System.Windows.Forms]System
        .Windows.Forms.RadioButton::set_Checked(bool)
1577    IL_01dd:  ldarg.0
```

```
1578      IL_01de:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1579      IL_01e3:  ldc.i4 0
1580      IL_01e8:  ldc.i4 20
1581      IL_01ed:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
1582      IL_01f2:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1583      IL_01f7:  ldloc.1
1584      IL_01f8:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1585      IL_01fd:  ldarg.0
1586      IL_01fe:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1587      IL_0203:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1588      IL_0208:  ldarg.0
1589      IL_0209:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.RadioButton::'.ctor'()
1590      IL_020e:  stfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio2
1591      IL_0213:  ldarg.0
1592      IL_0214:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio2
1593      IL_0219:  ldstr "Domme AI"
1594      IL_021e:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.ButtonBase::set_Text(string)
1595      IL_0223:  ldarg.0
1596      IL_0224:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio2
1597      IL_0229:  ldc.i4 0
1598      IL_022e:  ldc.i4 40
1599      IL_0233:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
1600      IL_0238:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1601      IL_023d:  ldloc.1
1602      IL_023e:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1603      IL_0243:  ldarg.0
1604      IL_0244:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio2
1605      IL_0249:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1606      IL_024e:  ldarg.0
1607      IL_024f:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.RadioButton::'.ctor'()
1608      IL_0254:  stfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio3
1609      IL_0259:  ldarg.0
1610      IL_025a:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio3
1611      IL_025f:  ldstr "Slimme AI"
1612      IL_0264:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.ButtonBase::set_Text(string)
1613      IL_0269:  ldarg.0
```

```
1614      IL_026a:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio3
1615      IL_026f:  ldc.i4 0
1616      IL_0274:  ldc.i4 60
1617      IL_0279:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
1618      IL_027e:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1619      IL_0283:  ldloc.1
1620      IL_0284:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1621      IL_0289:  ldarg.0
1622      IL_028a:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio3
1623      IL_028f:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1624      IL_0294:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.Panel::'.ctor'()
1625      IL_0299:  stloc.1
1626      IL_029a:  ldloc.1
1627      IL_029b:  ldc.i4 250
1628      IL_02a0:  ldc.i4 250
1629      IL_02a5:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Size::'.ctor'(int32, int32)
1630      IL_02aa:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Size(valuetype [System.Drawing]
              System.Drawing.Size)
1631      IL_02af:  ldloc.1
1632      IL_02b0:  ldc.i4 25
1633      IL_02b5:  ldc.i4 110
1634      IL_02ba:  newobj instance void valuetype [System.Drawing]System
              .Drawing.Point::'.ctor'(int32, int32)
1635      IL_02bf:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1636      IL_02c4:  ldarg.0
1637      IL_02c5:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1638      IL_02ca:  ldloc.1
1639      IL_02cb:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1640      IL_02d0:  ldarg.0
1641      IL_02d1:  ldc.i4 9
1642      IL_02d6:  newarr [System.Windows.Forms]System.Windows.Forms.
              Button
1643      IL_02db:  stfld class [System.Windows.Forms]System.Windows.
              Forms.Button[] BoterKaasEieren.BkeGui::tiles
1644      IL_02e0:  ldc.i4 0
1645      IL_02e5:  stloc.3
1646      IL_02e6:  ldloc.3
1647      IL_02e7:  ldc.i4 9
1648      IL_02ec:  clt
1649      IL_02ee:  brfalse IL_038d
1650
1651      IL_02f3:  newobj instance void class [System.Windows.Forms]
              System.Windows.Forms.Button::'.ctor'()
1652      IL_02f8:  stloc.s 4
```

```
1653        IL_02fa:  ldarg.0
1654        IL_02fb:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button[] BoterKaasEieren.BkeGui::tiles
1655        IL_0300:  ldloc.3
1656        IL_0301:  ldloc.s 4
1657        IL_0303:  stelem.any [System.Windows.Forms]System.Windows.Forms
                .Button
1658        IL_0308:  ldloc.s 4
1659        IL_030a:  ldloca.s 3
1660        IL_030c:  call instance string int32::ToString()
1661        IL_0311:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::set_Name(string)
1662        IL_0316:  ldloc.s 4
1663        IL_0318:  ldloca.s 3
1664        IL_031a:  call instance string int32::ToString()
1665        IL_031f:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.ButtonBase::set_Text(string)
1666        IL_0324:  ldloc.s 4
1667        IL_0326:  ldc.i4 40
1668        IL_032b:  ldc.i4 40
1669        IL_0330:  newobj instance void valuetype [System.Drawing]System
                .Drawing.Size::'.ctor'(int32, int32)
1670        IL_0335:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::set_Size(valuetype [System.Drawing]
                System.Drawing.Size)
1671        IL_033a:  ldloc.s 4
1672        IL_033c:  ldloc.3
1673        IL_033d:  ldc.i4 3
1674        IL_0342:  rem
1675        IL_0343:  ldc.i4 40
1676        IL_0348:  mul
1677        IL_0349:  ldloc.3
1678        IL_034a:  ldc.i4 3
1679        IL_034f:  div
1680        IL_0350:  ldc.i4 40
1681        IL_0355:  mul
1682        IL_0356:  newobj instance void valuetype [System.Drawing]System
                .Drawing.Point::'.ctor'(int32, int32)
1683        IL_035b:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::set_Location(valuetype [System.
                Drawing]System.Drawing.Point)
1684        IL_0360:  ldloc.s 4
1685        IL_0362:  ldarg.0
1686        IL_0363:  ldftn instance void class BoterKaasEieren.BkeGui::
                move_Clicked(object, class [mscorlib]System.EventArgs)
1687        IL_0369:  newobj instance void class [mscorlib]System.
                EventHandler::'.ctor'(object, native int)
1688        IL_036e:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::add_Click(class [mscorlib]System.
                EventHandler)
1689        IL_0373:  ldloc.1
1690        IL_0374:  call instance class [System.Windows.Forms]System.
                Windows.Forms.Control/ControlCollection class [System.
                Windows.Forms]System.Windows.Forms.Control::get_Controls()
1691        IL_0379:  ldloc.s 4
1692        IL_037b:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.Control/ControlCollection::Add(class [
                System.Windows.Forms]System.Windows.Forms.Control)
1693        IL_0380:  ldloc.3
1694        IL_0381:  ldc.i4 1
1695        IL_0386:  add
1696        IL_0387:  stloc.3
```

```
1697        IL_0388:  br IL_02e6
1698
1699        IL_038d:  ldarg.0
1700        IL_038e:  newobj instance void class [System.Windows.Forms]
                System.Windows.Forms.Button::'.ctor'()
1701        IL_0393:  stfld class [System.Windows.Forms]System.Windows.
                Forms.Button BoterKaasEieren.BkeGui::button
1702        IL_0398:  ldarg.0
1703        IL_0399:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button BoterKaasEieren.BkeGui::button
1704        IL_039e:  ldc.i4 180
1705        IL_03a3:  ldc.i4 20
1706        IL_03a8:  newobj instance void valuetype [System.Drawing]System
                .Drawing.Point::'.ctor'(int32, int32)
1707        IL_03ad:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::set_Location(valuetype [System.
                Drawing]System.Drawing.Point)
1708        IL_03b2:  ldarg.0
1709        IL_03b3:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button BoterKaasEieren.BkeGui::button
1710        IL_03b8:  ldstr "Start!"
1711        IL_03bd:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.ButtonBase::set_Text(string)
1712        IL_03c2:  ldarg.0
1713        IL_03c3:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button BoterKaasEieren.BkeGui::button
1714        IL_03c8:  ldarg.0
1715        IL_03c9:  ldftn instance void class BoterKaasEieren.BkeGui::
                button_Clicked(object, class [mscorlib]System.EventArgs)
1716        IL_03cf:  newobj instance void class [mscorlib]System.
                EventHandler::'.ctor'(object, native int)
1717        IL_03d4:  call instance void class [System.Windows.Forms]System
                .Windows.Forms.Control::add_Click(class [mscorlib]System.
                EventHandler)
1718        IL_03d9:  ldarg.0
1719        IL_03da:  call instance class [System.Windows.Forms]System.
                Windows.Forms.Control/ControlCollection class [System.
                Windows.Forms]System.Windows.Forms.Control::get_Controls()
1720        IL_03df:  ldarg.0
1721        IL_03e0:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button BoterKaasEieren.BkeGui::button
1722        IL_03e5:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.Control/ControlCollection::Add(class [
                System.Windows.Forms]System.Windows.Forms.Control)
1723        IL_03ea:  ldarg.0
1724        IL_03eb:  newobj instance void class [System.Windows.Forms]
                System.Windows.Forms.Label::'.ctor'()
1725        IL_03f0:  stfld class [System.Windows.Forms]System.Windows.
                Forms.Label BoterKaasEieren.BkeGui::label
1726        IL_03f5:  ldarg.0
1727        IL_03f6:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Label BoterKaasEieren.BkeGui::label
1728        IL_03fb:  ldstr "Press the start button!"
1729        IL_0400:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.Label::set_Text(string)
1730        IL_0405:  ldarg.0
1731        IL_0406:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Label BoterKaasEieren.BkeGui::label
1732        IL_040b:  ldc.i4 180
1733        IL_0410:  ldc.i4 50
1734        IL_0415:  newobj instance void valuetype [System.Drawing]System
                .Drawing.Point::'.ctor'(int32, int32)
```

```
1735      IL_041a:  call instance void class [System.Windows.Forms]System
              .Windows.Forms.Control::set_Location(valuetype [System.
              Drawing]System.Drawing.Point)
1736      IL_041f:  ldarg.0
1737      IL_0420:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.Label BoterKaasEieren.BkeGui::label
1738      IL_0425:  ldc.i4.1
1739      IL_0426:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Label::set_AutoSize(bool)
1740      IL_042b:  ldarg.0
1741      IL_042c:  call instance class [System.Windows.Forms]System.
              Windows.Forms.Control/ControlCollection class [System.
              Windows.Forms]System.Windows.Forms.Control::get_Controls()
1742      IL_0431:  ldarg.0
1743      IL_0432:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.Label BoterKaasEieren.BkeGui::label
1744      IL_0437:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Control/ControlCollection::Add(class [
              System.Windows.Forms]System.Windows.Forms.Control)
1745      IL_043c:  ret
1746      } // end of method BkeGui::.ctor
1747
1748      // method line 53
1749      .method public static  hidebysig
1750            default void Main ()  cil managed
1751      {
1752          // Method begins at RVA 0x2ecc
1753      .entrypoint
1754      // Code size 12 (0xc)
1755      .maxstack 2
1756      .locals init (
1757          object  V_0)
1758      IL_0000:  nop
1759      IL_0001:  newobj instance void class BoterKaasEieren.BkeGui::'.
              ctor'()
1760      IL_0006:  call void class [System.Windows.Forms]System.Windows.
              Forms.Application::Run(class [System.Windows.Forms]System.
              Windows.Forms.Form)
1761      IL_000b:  ret
1762      } // end of method BkeGui::Main
1763
1764      // method line 54
1765      .method public virtual  hidebysig  newslot
1766            instance default void MessageReceived (object message)
                  cil managed
1767      {
1768          // Method begins at RVA 0x2ee4
1769      // Code size 353 (0x161)
1770      .maxstack 34
1771      .locals init (
1772          object  V_0,
1773          int32   V_1,
1774          class BoterKaasEieren.Speler    V_2,
1775          class BoterKaasEieren.Bord  V_3,
1776          int32   V_4,
1777          class BoterKaasEieren.Speler    V_5,
1778          int32   V_6,
1779          class BoterKaasEieren.Speler    V_7)
1780      IL_0000:  nop
1781      IL_0001:  ldarg 1
1782
1783      IL_0005:  ldnull
```

```
1784        IL_0006:  ceq
1785        IL_0008:  brfalse IL_0071
1786
1787        IL_000d:  ldc.i4 0
1788        IL_0012:  stloc.1
1789        IL_0013:  ldloc.1
1790        IL_0014:  ldc.i4 9
1791        IL_0019:  clt
1792        IL_001b:  brfalse IL_0045
1793
1794        IL_0020:  ldarg.0
1795        IL_0021:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.Button[] BoterKaasEieren.BkeGui::tiles
1796        IL_0026:  ldloc.1
1797        IL_0027:  ldelem.any [System.Windows.Forms]System.Windows.Forms
              .Button
1798        IL_002c:  ldloca.s 1
1799        IL_002e:  call instance string int32::ToString()
1800        IL_0033:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.ButtonBase::set_Text(string)
1801        IL_0038:  ldloc.1
1802        IL_0039:  ldc.i4 1
1803        IL_003e:  add
1804        IL_003f:  stloc.1
1805        IL_0040:  br IL_0013
1806
1807        IL_0045:  ldarg.0
1808        IL_0046:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
1809        IL_004b:  call instance class BoterKaasEieren.Speler class
              BoterKaasEieren.Spel::GetHuidigeSpeler()
1810        IL_0050:  stloc.2
1811        IL_0051:  ldarg.0
1812        IL_0052:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.Label BoterKaasEieren.BkeGui::label
1813        IL_0057:  ldloc.2
1814        IL_0058:  call instance string class BoterKaasEieren.Speler::
              GetNaam()
1815        IL_005d:  ldstr " is aan zet!"
1816        IL_0062:  call string string::Concat(object, object)
1817        IL_0067:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Label::set_Text(string)
1818        IL_006c:  br IL_0160
1819
1820        IL_0071:  ldarg 1
1821
1822        IL_0075:  isinst BoterKaasEieren.Bord
1823        IL_007a:  ldnull
1824        IL_007b:  cgt.un
1825        IL_007d:  brfalse IL_00fd
1826
1827        IL_0082:  ldarg 1
1828
1829        IL_0086:  castclass BoterKaasEieren.Bord
1830        IL_008b:  stloc.3
1831        IL_008c:  ldc.i4 0
1832        IL_0091:  stloc.s 4
1833        IL_0093:  ldloc.s 4
1834        IL_0095:  ldc.i4 9
1835        IL_009a:  clt
1836        IL_009c:  brfalse IL_00cf
1837
```

```
1838        IL_00a1:  ldarg.0
1839        IL_00a2:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Button[] BoterKaasEieren.BkeGui::tiles
1840        IL_00a7:  ldloc.s 4
1841        IL_00a9:  ldelem.any [System.Windows.Forms]System.Windows.Forms
                .Button
1842        IL_00ae:  ldloc.3
1843        IL_00af:  ldloc.s 4
1844        IL_00b1:  call instance class BoterKaasEieren.Mark class
                BoterKaasEieren.Bord::GetVakje(int32)
1845        IL_00b6:  callvirt instance string class BoterKaasEieren.Mark::
                ToString()
1846        IL_00bb:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.ButtonBase::set_Text(string)
1847        IL_00c0:  ldloc.s 4
1848        IL_00c2:  ldc.i4 1
1849        IL_00c7:  add
1850        IL_00c8:  stloc.s 4
1851        IL_00ca:  br IL_0093
1852
1853        IL_00cf:  ldarg.0
1854        IL_00d0:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
                BkeGui::spel
1855        IL_00d5:  call instance class BoterKaasEieren.Speler class
                BoterKaasEieren.Spel::GetHuidigeSpeler()
1856        IL_00da:  stloc.s 5
1857        IL_00dc:  ldarg.0
1858        IL_00dd:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Label BoterKaasEieren.BkeGui::label
1859        IL_00e2:  ldloc.s 5
1860        IL_00e4:  call instance string class BoterKaasEieren.Speler::
                GetNaam()
1861        IL_00e9:  ldstr " is aan zet!"
1862        IL_00ee:  call string string::Concat(object, object)
1863        IL_00f3:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.Label::set_Text(string)
1864        IL_00f8:  br IL_0160
1865
1866        IL_00fd:  ldarg 1
1867
1868        IL_0101:  unbox [mscorlib]System.Int32
1869        IL_0106:  ldobj [mscorlib]System.Int32
1870        IL_010b:  stloc.s 6
1871        IL_010d:  ldloc.s 6
1872        IL_010f:  ldc.i4 -1
1873        IL_0114:  ceq
1874        IL_0116:  brfalse IL_0130
1875
1876        IL_011b:  ldarg.0
1877        IL_011c:  ldfld class [System.Windows.Forms]System.Windows.
                Forms.Label BoterKaasEieren.BkeGui::label
1878        IL_0121:  ldstr "Gelijkspel!"
1879        IL_0126:  callvirt instance void class [System.Windows.Forms]
                System.Windows.Forms.Label::set_Text(string)
1880        IL_012b:  br IL_0160
1881
1882        IL_0130:  ldarg.0
1883        IL_0131:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
                BkeGui::spel
1884        IL_0136:  call instance class BoterKaasEieren.Speler[] class
                BoterKaasEieren.Spel::GetSpelers()
1885        IL_013b:  ldloc.s 6
```

```
1886      IL_013d:  ldelem.any BoterKaasEieren.Speler
1887      IL_0142:  stloc.s 7
1888      IL_0144:  ldarg.0
1889      IL_0145:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.Label BoterKaasEieren.BkeGui::label
1890      IL_014a:  ldloc.s 7
1891      IL_014c:  call instance string class BoterKaasEieren.Speler::
              GetNaam()
1892      IL_0151:  ldstr " heeft gewonnen!"
1893      IL_0156:  call string string::Concat(object, object)
1894      IL_015b:  callvirt instance void class [System.Windows.Forms]
              System.Windows.Forms.Label::set_Text(string)
1895      IL_0160:  ret
1896      } // end of method BkeGui::MessageReceived
1897
1898      // method line 55
1899      .method private hidebysig
1900            instance default void button_Clicked (object sender,
                  class [mscorlib]System.EventArgs e)  cil managed
1901      {
1902          // Method begins at RVA 0x3054
1903      // Code size 289 (0x121)
1904      .maxstack 18
1905      .locals init (
1906          object  V_0,
1907          class BoterKaasEieren.Speler    V_1,
1908          class BoterKaasEieren.Speler    V_2,
1909          class [mscorlib]System.Threading.Thread V_3)
1910      IL_0000:  nop
1911      IL_0001:  ldarg.0
1912      IL_0002:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio1
1913      IL_0007:  call instance bool class [System.Windows.Forms]System
              .Windows.Forms.RadioButton::get_Checked()
1914      IL_000c:  brfalse IL_002a
1915
1916      IL_0011:  ldstr "Speler 1"
1917      IL_0016:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::XX
1918      IL_001b:  newobj instance void class BoterKaasEieren.MensSpeler
              ::'.ctor'(string, class BoterKaasEieren.Mark)
1919      IL_0020:  castclass BoterKaasEieren.Speler
1920      IL_0025:  br IL_0071
1921
1922      IL_002a:  ldarg.0
1923      IL_002b:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s1radio2
1924      IL_0030:  call instance bool class [System.Windows.Forms]System
              .Windows.Forms.RadioButton::get_Checked()
1925      IL_0035:  brfalse IL_0058
1926
1927      IL_003a:  ldstr "Speler 1"
1928      IL_003f:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::XX
1929      IL_0044:  newobj instance void class BoterKaasEieren.
              DommeStrategie::'.ctor'()
1930      IL_0049:  newobj instance void class BoterKaasEieren.
              ComputerSpeler::'.ctor'(string, class BoterKaasEieren.Mark,
               class BoterKaasEieren.IStrategie)
1931      IL_004e:  castclass BoterKaasEieren.Speler
1932      IL_0053:  br IL_0071
1933
```

```
1934      IL_0058:  ldstr "Speler 1"
1935      IL_005d:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::XX
1936      IL_0062:  newobj instance void class BoterKaasEieren.
              BesteStrategie::'.ctor'()
1937      IL_0067:  newobj instance void class BoterKaasEieren.
              ComputerSpeler::'.ctor'(string, class BoterKaasEieren.Mark,
               class BoterKaasEieren.IStrategie)
1938      IL_006c:  castclass BoterKaasEieren.Speler
1939      IL_0071:  stloc.1
1940      IL_0072:  ldarg.0
1941      IL_0073:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio1
1942      IL_0078:  call instance bool class [System.Windows.Forms]System
              .Windows.Forms.RadioButton::get_Checked()
1943      IL_007d:  brfalse IL_009b
1944
1945      IL_0082:  ldstr "Speler 2"
1946      IL_0087:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::OO
1947      IL_008c:  newobj instance void class BoterKaasEieren.MensSpeler
              ::'.ctor'(string, class BoterKaasEieren.Mark)
1948      IL_0091:  castclass BoterKaasEieren.Speler
1949      IL_0096:  br IL_00e2
1950
1951      IL_009b:  ldarg.0
1952      IL_009c:  ldfld class [System.Windows.Forms]System.Windows.
              Forms.RadioButton BoterKaasEieren.BkeGui::s2radio2
1953      IL_00a1:  call instance bool class [System.Windows.Forms]System
              .Windows.Forms.RadioButton::get_Checked()
1954      IL_00a6:  brfalse IL_00c9
1955
1956      IL_00ab:  ldstr "Speler 2"
1957      IL_00b0:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::OO
1958      IL_00b5:  newobj instance void class BoterKaasEieren.
              DommeStrategie::'.ctor'()
1959      IL_00ba:  newobj instance void class BoterKaasEieren.
              ComputerSpeler::'.ctor'(string, class BoterKaasEieren.Mark,
               class BoterKaasEieren.IStrategie)
1960      IL_00bf:  castclass BoterKaasEieren.Speler
1961      IL_00c4:  br IL_00e2
1962
1963      IL_00c9:  ldstr "Speler 2"
1964      IL_00ce:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::OO
1965      IL_00d3:  newobj instance void class BoterKaasEieren.
              BesteStrategie::'.ctor'()
1966      IL_00d8:  newobj instance void class BoterKaasEieren.
              ComputerSpeler::'.ctor'(string, class BoterKaasEieren.Mark,
               class BoterKaasEieren.IStrategie)
1967      IL_00dd:  castclass BoterKaasEieren.Speler
1968      IL_00e2:  stloc.2
1969      IL_00e3:  ldarg.0
1970      IL_00e4:  ldloc.1
1971      IL_00e5:  ldloc.2
1972      IL_00e6:  newobj instance void class BoterKaasEieren.Spel::'.
              ctor'(class BoterKaasEieren.Speler, class BoterKaasEieren.
              Speler)
1973      IL_00eb:  stfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
1974      IL_00f0:  ldarg.0
```

```
1975      IL_00f1:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
1976      IL_00f6:  ldarg.0
1977      IL_00f7:  call instance void class BoterKaasEieren.Spel::
              AddObserver(class BoterKaasEieren.IMessageListener)
1978      IL_00fc:  ldarg.0
1979      IL_00fd:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
1980      IL_0102:  ldftn instance void class BoterKaasEieren.Spel::Play
              ()
1981      IL_0108:  newobj instance void class [mscorlib]System.Threading
              .ThreadStart::'.ctor'(object, native int)
1982      IL_010d:  newobj instance void class [mscorlib]System.Threading
              .Thread::'.ctor'(class [mscorlib]System.Threading.
              ThreadStart)
1983      IL_0112:  stloc.3
1984      IL_0113:  ldloc.3
1985      IL_0114:  ldc.i4.1
1986      IL_0115:  call instance void class [mscorlib]System.Threading.
              Thread::set_IsBackground(bool)
1987      IL_011a:  ldloc.3
1988      IL_011b:  call instance void class [mscorlib]System.Threading.
              Thread::Start()
1989      IL_0120:  ret
1990      } // end of method BkeGui::button_Clicked
1991
1992      // method line 56
1993      .method private hidebysig
1994            instance default void move_Clicked (object sender, class
                  [mscorlib]System.EventArgs e)  cil managed
1995      {
1996          // Method begins at RVA 0x3184
1997      // Code size 90 (0x5a)
1998      .maxstack 13
1999      .locals init (
2000          object  V_0,
2001          class [System.Windows.Forms]System.Windows.Forms.Button V_1
                  ,
2002          int32   V_2,
2003          class BoterKaasEieren.MensSpeler    V_3)
2004      IL_0000:  nop
2005      IL_0001:  ldarg.0
2006      IL_0002:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
2007      IL_0007:  call instance class BoterKaasEieren.Speler class
              BoterKaasEieren.Spel::GetHuidigeSpeler()
2008      IL_000c:  isinst BoterKaasEieren.MensSpeler
2009      IL_0011:  ldnull
2010      IL_0012:  cgt.un
2011      IL_0014:  brfalse IL_0059
2012
2013      IL_0019:  ldarg 1
2014
2015      IL_001d:  castclass [System.Windows.Forms]System.Windows.Forms.
              Button
2016      IL_0022:  stloc.1
2017      IL_0023:  ldloc.1
2018      IL_0024:  call instance string class [System.Windows.Forms]
              System.Windows.Forms.Control::get_Name()
2019      IL_0029:  call int32 int32::Parse(string)
2020      IL_002e:  stloc.2
2021      IL_002f:  ldarg.0
```

```
2022      IL_0030:  ldfld class BoterKaasEieren.Spel BoterKaasEieren.
              BkeGui::spel
2023      IL_0035:  call instance class BoterKaasEieren.Speler class
              BoterKaasEieren.Spel::GetHuidigeSpeler()
2024      IL_003a:  castclass BoterKaasEieren.MensSpeler
2025      IL_003f:  stloc.3
2026      IL_0040:  ldloc.3
2027      IL_0041:  call void class [mscorlib]System.Threading.Monitor::
              Enter(object)
2028      IL_0046:  ldloc.3
2029      IL_0047:  ldloc.2
2030      IL_0048:  call instance void class BoterKaasEieren.MensSpeler::
              SetTempMove(int32)
2031      IL_004d:  ldloc.3
2032      IL_004e:  call void class [mscorlib]System.Threading.Monitor::
              Pulse(object)
2033      IL_0053:  ldloc.3
2034      IL_0054:  call void class [mscorlib]System.Threading.Monitor::
              Exit(object)
2035      IL_0059:  ret
2036      } // end of method BkeGui::move_Clicked
2037
2038    } // end of class BoterKaasEieren.BkeGui
2039  }
2040
2041  .namespace BoterKaasEieren
2042  {
2043    .class interface public auto ansi abstract IMessageListener
2044      implements [System]System.ComponentModel.ISynchronizeInvoke  {
2045
2046      // method line 57
2047      .method public virtual  hidebysig  newslot  abstract
2048              instance default void MessageReceived (object message)
                    cil managed
2049      {
2050          // Method begins at RVA 0x0
2051      } // end of method IMessageListener::MessageReceived
2052
2053    } // end of class BoterKaasEieren.IMessageListener
2054  }
2055
2056  .namespace BoterKaasEieren
2057  {
2058    .class public auto ansi BesteStrategie
2059      extends [mscorlib]System.Object
2060      implements BoterKaasEieren.IStrategie  {
2061      .field  private  class BoterKaasEieren.Mark dezeSpeler
2062      .field  private  class [mscorlib]System.Random random
2063
2064      // method line 58
2065      .method public hidebysig  specialname  rtspecialname
2066              instance default void '.ctor' ()  cil managed
2067      {
2068          // Method begins at RVA 0x31ec
2069      // Code size 18 (0x12)
2070      .maxstack 4
2071      .locals init (
2072          object  V_0)
2073      IL_0000:  ldarg.0
2074      IL_0001:  call instance void object::'.ctor'()
2075      IL_0006:  ldarg.0
```

```
2076      IL_0007:  newobj instance void class [mscorlib]System.Random
              ::'.ctor'()
2077      IL_000c:  stfld class [mscorlib]System.Random BoterKaasEieren.
              BesteStrategie::random
2078      IL_0011:  ret
2079      } // end of method BesteStrategie::.ctor
2080
2081      // method line 59
2082      .method public virtual  hidebysig  newslot
2083              instance default int32 BerekenZet (class BoterKaasEieren
                    .Bord bord, class BoterKaasEieren.Mark mark)  cil
                    managed
2084      {
2085          // Method begins at RVA 0x320c
2086      // Code size 31 (0x1f)
2087      .maxstack 4
2088      .locals init (
2089          object  V_0)
2090      IL_0000:  nop
2091      IL_0001:  ldarg.0
2092      IL_0002:  ldarg 2
2093
2094      IL_0006:  stfld class BoterKaasEieren.Mark BoterKaasEieren.
              BesteStrategie::dezeSpeler
2095      IL_000b:  ldarg.0
2096      IL_000c:  ldarg 1
2097
2098      IL_0010:  ldarg 2
2099
2100      IL_0014:  call instance class BoterKaasEieren.Zet class
              BoterKaasEieren.BesteStrategie::_berekenZet(class
              BoterKaasEieren.Bord, class BoterKaasEieren.Mark)
2101      IL_0019:  ldfld int32 BoterKaasEieren.Zet::zet
2102      IL_001e:  ret
2103      } // end of method BesteStrategie::BerekenZet
2104
2105      // method line 60
2106      .method private hidebysig
2107              instance default class BoterKaasEieren.Zet _berekenZet (
                    class BoterKaasEieren.Bord bord, class
                    BoterKaasEieren.Mark mark)  cil managed
2108      {
2109          // Method begins at RVA 0x3238
2110      // Code size 387 (0x183)
2111      .maxstack 23
2112      .locals init (
2113          object  V_0,
2114          bool    V_1,
2115          class BoterKaasEieren.Zet   V_2,
2116          int32   V_3,
2117          int32   V_4,
2118          int32   V_5,
2119          class BoterKaasEieren.Zet   V_6)
2120      IL_0000:  nop
2121      IL_0001:  ldc.i4.1
2122      IL_0002:  stloc.1
2123      IL_0003:  ldnull
2124      IL_0004:  stloc.2
2125      IL_0005:  ldc.i4 0
2126      IL_000a:  stloc.3
2127      IL_000b:  ldarg.0
```

```
2128      IL_000c:  ldfld class [mscorlib]System.Random BoterKaasEieren.
              BesteStrategie::random
2129      IL_0011:  ldsfld int32 BoterKaasEieren.Bord::DIM
2130      IL_0016:  ldsfld int32 BoterKaasEieren.Bord::DIM
2131      IL_001b:  mul
2132      IL_001c:  callvirt instance int32 class [mscorlib]System.Random
              ::Next(int32)
2133      IL_0021:  stloc.s 4
2134      IL_0023:  ldloc.3
2135      IL_0024:  ldsfld int32 BoterKaasEieren.Bord::DIM
2136      IL_0029:  ldsfld int32 BoterKaasEieren.Bord::DIM
2137      IL_002e:  mul
2138      IL_002f:  clt
2139      IL_0031:  brfalse IL_003c
2140
2141      IL_0036:  ldloc.1
2142      IL_0037:  br IL_003d
2143
2144      IL_003c:  ldc.i4.0
2145      IL_003d:  brfalse IL_0181
2146
2147      IL_0042:  ldloc.3
2148      IL_0043:  ldloc.s 4
2149      IL_0045:  add
2150      IL_0046:  ldsfld int32 BoterKaasEieren.Bord::DIM
2151      IL_004b:  ldsfld int32 BoterKaasEieren.Bord::DIM
2152      IL_0050:  mul
2153      IL_0051:  rem
2154      IL_0052:  stloc.s 5
2155      IL_0054:  ldarg 1
2156
2157      IL_0058:  ldloc.s 5
2158      IL_005a:  call instance bool class BoterKaasEieren.Bord::
              IsLeegVakje(int32)
2159      IL_005f:  brfalse IL_0174
2160
2161      IL_0064:  newobj instance void class BoterKaasEieren.Zet::'.
              ctor'()
2162      IL_0069:  stloc.s 6
2163      IL_006b:  ldloc.s 6
2164      IL_006d:  ldloc.s 5
2165      IL_006f:  stfld int32 BoterKaasEieren.Zet::zet
2166      IL_0074:  ldarg 1
2167
2168      IL_0078:  ldloc.s 5
2169      IL_007a:  ldarg 2
2170
2171      IL_007e:  call instance void class BoterKaasEieren.Bord::
              SetVakje(int32, class BoterKaasEieren.Mark)
2172      IL_0083:  ldarg 1
2173
2174      IL_0087:  call instance bool class BoterKaasEieren.Bord::
              GameOver()
2175      IL_008c:  brfalse IL_00ff
2176
2177      IL_0091:  ldarg 1
2178
2179      IL_0095:  ldarg 2
2180
2181      IL_0099:  call instance bool class BoterKaasEieren.Bord::
              HeeftGewonnen(class BoterKaasEieren.Mark)
2182      IL_009e:  brfalse IL_00d2
```

```
2183
2184        IL_00a3:  ldc.i4.0
2185        IL_00a4:  stloc.1
2186        IL_00a5:  ldloc.s 6
2187        IL_00a7:  ldc.i4 1
2188        IL_00ac:  box [mscorlib]System.Int32
2189        IL_00b1:  stloc.0
2190        IL_00b2:  ldloc.0
2191        IL_00b3:  unbox [mscorlib]System.Int32
2192        IL_00b8:  ldobj [mscorlib]System.Int32
2193        IL_00bd:  stfld int32 BoterKaasEieren.Zet::waarde
2194        IL_00c2:  ldloc.0
2195        IL_00c3:  unbox [mscorlib]System.Int32
2196        IL_00c8:  ldobj [mscorlib]System.Int32
2197        IL_00cd:  br IL_00fa
2198
2199        IL_00d2:  ldloc.s 6
2200        IL_00d4:  ldc.i4 0
2201        IL_00d9:  box [mscorlib]System.Int32
2202        IL_00de:  stloc.0
2203        IL_00df:  ldloc.0
2204        IL_00e0:  unbox [mscorlib]System.Int32
2205        IL_00e5:  ldobj [mscorlib]System.Int32
2206        IL_00ea:  stfld int32 BoterKaasEieren.Zet::waarde
2207        IL_00ef:  ldloc.0
2208        IL_00f0:  unbox [mscorlib]System.Int32
2209        IL_00f5:  ldobj [mscorlib]System.Int32
2210        IL_00fa:  br IL_013b
2211
2212        IL_00ff:  ldloc.s 6
2213        IL_0101:  ldarg.0
2214        IL_0102:  ldarg 1
2215
2216        IL_0106:  ldarg 2
2217
2218        IL_010a:  call instance class BoterKaasEieren.Mark class
              BoterKaasEieren.Mark::Other()
2219        IL_010f:  call instance class BoterKaasEieren.Zet class
              BoterKaasEieren.BesteStrategie::_berekenZet(class
              BoterKaasEieren.Bord, class BoterKaasEieren.Mark)
2220        IL_0114:  ldfld int32 BoterKaasEieren.Zet::waarde
2221        IL_0119:  neg
2222        IL_011a:  box [mscorlib]System.Int32
2223        IL_011f:  stloc.0
2224        IL_0120:  ldloc.0
2225        IL_0121:  unbox [mscorlib]System.Int32
2226        IL_0126:  ldobj [mscorlib]System.Int32
2227        IL_012b:  stfld int32 BoterKaasEieren.Zet::waarde
2228        IL_0130:  ldloc.0
2229        IL_0131:  unbox [mscorlib]System.Int32
2230        IL_0136:  ldobj [mscorlib]System.Int32
2231        IL_013b:  pop
2232        IL_013c:  ldloc.2
2233        IL_013d:  ldnull
2234        IL_013e:  ceq
2235        IL_0140:  brtrue IL_0159
2236
2237        IL_0145:  ldloc.s 6
2238        IL_0147:  ldfld int32 BoterKaasEieren.Zet::waarde
2239        IL_014c:  ldloc.2
2240        IL_014d:  ldfld int32 BoterKaasEieren.Zet::waarde
2241        IL_0152:  cgt
```

```
2242      IL_0154:  br IL_015a
2243
2244      IL_0159:  ldc.i4.1
2245      IL_015a:  brfalse IL_0164
2246
2247      IL_015f:  ldloc.s 6
2248      IL_0161:  stloc.2
2249      IL_0162:  ldloc.2
2250      IL_0163:  pop
2251      IL_0164:  ldarg 1
2252
2253      IL_0168:  ldloc.s 5
2254      IL_016a:  ldsfld class BoterKaasEieren.Mark BoterKaasEieren.
              Mark::EMPTY
2255      IL_016f:  call instance void class BoterKaasEieren.Bord::
              SetVakje(int32, class BoterKaasEieren.Mark)
2256      IL_0174:  ldloc.3
2257      IL_0175:  ldc.i4 1
2258      IL_017a:  add
2259      IL_017b:  stloc.3
2260      IL_017c:  br IL_0023
2261
2262      IL_0181:  ldloc.2
2263      IL_0182:  ret
2264      } // end of method BesteStrategie::_berekenZet
2265
2266    } // end of class BoterKaasEieren.BesteStrategie
2267  }
2268
2269  .namespace BoterKaasEieren
2270  {
2271    .class public auto ansi Zet
2272      extends [mscorlib]System.Object
2273    {
2274      .field  public  int32 zet
2275      .field  public  int32 waarde
2276
2277      // method line 61
2278      .method public hidebysig  specialname  rtspecialname
2279              instance default void '.ctor' ()  cil managed
2280      {
2281          // Method begins at RVA 0x33c8
2282      // Code size 7 (0x7)
2283      .maxstack 8
2284      IL_0000:  ldarg.0
2285      IL_0001:  call instance void object::'.ctor'()
2286      IL_0006:  ret
2287      } // end of method Zet::.ctor
2288
2289    } // end of class BoterKaasEieren.Zet
2290  }
2291
2292  .namespace BoterKaasEieren
2293  {
2294    .class public auto ansi DommeStrategie
2295      extends [mscorlib]System.Object
2296      implements BoterKaasEieren.IStrategie  {
2297      .field  private  class [mscorlib]System.Random random
2298
2299      // method line 62
2300      .method public hidebysig  specialname  rtspecialname
2301              instance default void '.ctor' ()  cil managed
```

```
2302      {
2303          // Method begins at RVA 0x33d0
2304      // Code size 18 (0x12)
2305      .maxstack 4
2306      .locals init (
2307          object  V_0)
2308      IL_0000:  ldarg.0
2309      IL_0001:  call instance void object::'.ctor'()
2310      IL_0006:  ldarg.0
2311      IL_0007:  newobj instance void class [mscorlib]System.Random
              ::'.ctor'()
2312      IL_000c:  stfld class [mscorlib]System.Random BoterKaasEieren.
              DommeStrategie::random
2313      IL_0011:  ret
2314      } // end of method DommeStrategie::.ctor
2315
2316      // method line 63
2317      .method public virtual  hidebysig  newslot
2318              instance default int32 BerekenZet (class BoterKaasEieren
                    .Bord bord, class BoterKaasEieren.Mark mark)  cil
                    managed
2319      {
2320          // Method begins at RVA 0x33f0
2321      // Code size 112 (0x70)
2322      .maxstack 12
2323      .locals init (
2324          object  V_0,
2325          class [mscorlib]System.Collections.ArrayList    V_1,
2326          int32   V_2)
2327      IL_0000:  nop
2328      IL_0001:  newobj instance void class [mscorlib]System.
              Collections.ArrayList::'.ctor'()
2329      IL_0006:  stloc.1
2330      IL_0007:  ldc.i4 0
2331      IL_000c:  stloc.2
2332      IL_000d:  ldloc.2
2333      IL_000e:  ldsfld int32 BoterKaasEieren.Bord::DIM
2334      IL_0013:  ldsfld int32 BoterKaasEieren.Bord::DIM
2335      IL_0018:  mul
2336      IL_0019:  clt
2337      IL_001b:  brfalse IL_0049
2338
2339      IL_0020:  ldarg 1
2340
2341      IL_0024:  ldloc.2
2342      IL_0025:  call instance bool class BoterKaasEieren.Bord::
              IsLeegVakje(int32)
2343      IL_002a:  brfalse IL_003c
2344
2345      IL_002f:  ldloc.1
2346      IL_0030:  ldloc.2
2347      IL_0031:  box [mscorlib]System.Int32
2348      IL_0036:  callvirt instance int32 class [mscorlib]System.
              Collections.ArrayList::Add(object)
2349      IL_003b:  pop
2350      IL_003c:  ldloc.2
2351      IL_003d:  ldc.i4 1
2352      IL_0042:  add
2353      IL_0043:  stloc.2
2354      IL_0044:  br IL_000d
2355
2356      IL_0049:  ldloc.1
```

```
2357        IL_004a:  ldarg.0
2358        IL_004b:  ldfld class [mscorlib]System.Random BoterKaasEieren.
                DommeStrategie::random
2359        IL_0050:  ldc.i4 0
2360        IL_0055:  ldloc.1
2361        IL_0056:  callvirt instance int32 class [mscorlib]System.
                Collections.ArrayList::get_Count()
2362        IL_005b:  callvirt instance int32 class [mscorlib]System.Random
                ::Next(int32, int32)
2363        IL_0060:  callvirt instance object class [mscorlib]System.
                Collections.ArrayList::get_Item(int32)
2364        IL_0065:  unbox [mscorlib]System.Int32
2365        IL_006a:  ldobj [mscorlib]System.Int32
2366        IL_006f:  ret
2367        } // end of method DommeStrategie::BerekenZet
2368
2369    } // end of class BoterKaasEieren.DommeStrategie
2370 }
```

F.3 Gebruiksaanwijzing

Aangezien *Boter, Kaas en Eieren* een *GUI* applicatie is valt er niet te spreken over gewone invoer en uitvoer. De applicatie is immers te bedienen met de muis. Om toch een beeld te geven van de werking van de applicatie en hoe de correctheid hiervan getest kan worden is er een korte gebruiksaanwijzing opgesteld.

Nadat de applicatie is opgestart kan voor beide spelers gekozen worden of hier een mens, een domme *AI* of een slimme *AI* speelt. Als deze keuze volledig is ingevuld kan een spel gestart worden met de start knop. Vervolgens kunnen de spelers om en om hun zet doen, totdat het spel afgelopen is. Een nieuw spel kan wederom gestart worden op dezelfde manier.