



ΜΕΤΑΦΡΑΣΤΗΣ SIMPLE

ΜΥΥ802 Μεταφραστές

Παναγιώτης Κολοκούρης
4914

1. Εισαγωγή

Αυτή η αναφορά έχει σκοπό να περιγράψει την ανάπτυξη, σε γλώσσα Python, ενός μεταφραστή για την γλώσσα προγραμματισμού *cimple*.

Η γλώσσα προγραμματισμού *cimple* είναι μια εκπαιδευτική γλώσσα προγραμματισμού που περιγράφεται στο έγγραφο «Η Γλώσσα Προγραμματισμού *cimple*».

Ο μεταφραστής που αναπτύχθηκε περιλαμβάνει τα εξής επιμέρους εργαλεία:

- Αναγνώστης κειμένου (file parser).
- Λεκτικός αναλυτής.
- Συντακτικός αναλυτής.
- Εργαλείο παραγωγής ενδιάμεσου κώδικα.
- Εργαλείο παραγωγής πίνακα συμβόλων.
- Εργαλείο παραγωγής τελικού κώδικα σε assembly.

Η λειτουργικότητα του μεταφραστή της *cimple* περιγράφεται εν συντομία στα επόμενα βήματα:

- Ο μεταφραστής διαβάζει ένα αρχείο κειμένου με πηγαίο κώδικα της γλώσσας *cimple*.
- Διατρέχει τον πηγαίο κώδικα και πραγματοποιεί λεκτική και συντακτική ανάλυση.
- Παράλληλα παράγει τον ενδιάμεσο κώδικα για κάθε μια από τις απαραίτητες εντολές πηγαίου κώδικα. Ο ενδιάμεσος κώδικας αποθηκεύεται στο αρχείο *test.int*.
- Δημιουργεί και κρατά στην μνήμη τον πίνακα συμβόλων για κάθε περιοχή (scope) του κώδικα. Για λόγους από-σφαλμάτωσης (debugging), ο πίνακας συμβόλων αποθηκεύεται στο αρχείο *test.symb*.
- Για λόγους από-σφαλμάτωσης, ο μεταφραστής παράγει από τον ενδιάμεσο κώδικα, ισοδύναμο πρόγραμμα σε γλώσσα c. Αποθηκεύεται στο αρχείο *test.c*.
- Τέλος, ο μεταφραστής παράγει τον τελικό κώδικα σε assembly για κάθε περιοχή (scope) ξεχωριστά όταν ολοκληρώνεται η αντίστοιχη μετάφραση της περιοχής. Ο τελικός κώδικας βασίζεται στον ενδιάμεσο κώδικα και στον πίνακα συμβόλων. Ο τελικός κώδικας αποθηκεύεται στο αρχείο *test.asm*.

Ομάδα Ανάπτυξης

Η ομάδα ανάπτυξης του μεταφραστή αποτελείται από τους εξής προγραμματιστές:

- Παναγιώτης Κολοκούρης, 4914

Δομή Αναφοράς

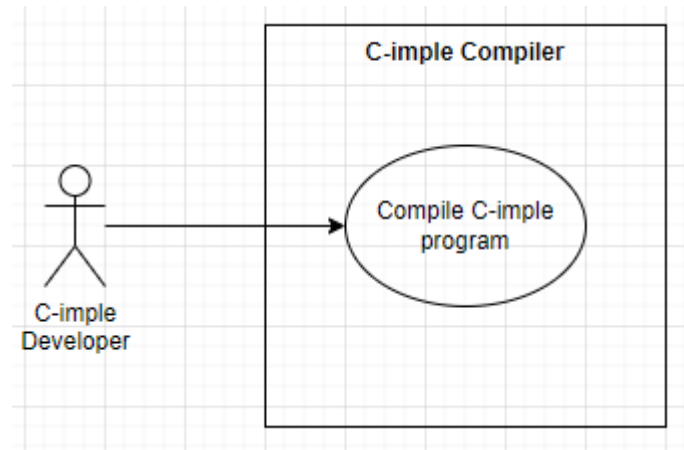
Η παρούσα αναφορά ακολουθεί την εξής δομή:

- Κεφάλαιο 1. Εισαγωγή
- Κεφάλαιο 2. Περιγράφει το use case που ικανοποιεί ο μεταφραστής.
- Κεφάλαιο 3. Περιγράφει την ανάπτυξη του μεταφραστή.

2. Use Cases

Η παρούσα ενότητα περιγράφει το use case που καλείται να ικανοποιήσει ο μεταφραστής, ήτοι την μεταγλώττιση ενός προγράμματος *cimple*. Στην Εικόνα 2.1 παρουσιάζεται το διάγραμμα των use cases.

Εικόνα 2.1: Διάγραμμα Use Cases



Το ροή του use case παρουσιάζεται στον Πίνακα 2.1.

Πίνακας 2.1: Use Case - Μετάφραση Προγράμματος Cimple

Use case ID	UC1
Actors	Cimple Developer
Preconditions	n/a
Main flow of events	<ol style="list-style-type: none">1. Ο προγραμματιστής καλεί από την γραμμή εντολών τον μεταφραστή παρέχοντας σαν όρισμα το όνομα του αρχείου, σε γλώσσα cimple, προς μετάφραση.2. Το σύστημα εκτελεί τον μεταφραστή.3. Το σύστημα εμφανίζει μήνυμα επιτυχίας στον προγραμματιστή.4. Το σύστημα παράγει και αποθηκεύει στον δίσκο (στην τοποθεσία από την οποία κλήθηκε) τα αρχεία <i>test.int</i>, <i>test.symb</i>, <i>test.c</i> και <i>test.asm</i>.
Alternative flow	Η μετάφραση αποτυγχάνει και το σύστημα εμφανίζει κατάλληλο πληροφοριακό μήνυμα στον προγραμματιστή με την πιθανή αιτία του λόγου αποτυχίας.
Post conditions	n/a

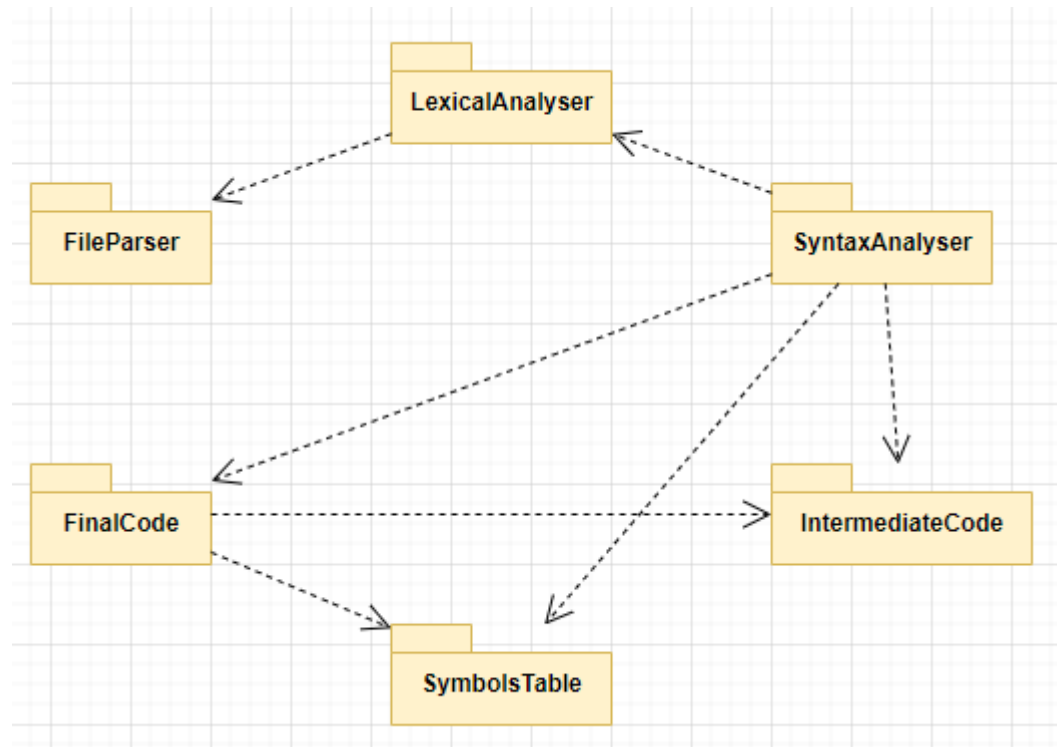
3. Ανάπτυξη Μεταφραστή

Η παρούσα ενότητα παρουσιάζει την λεπτομερή ανάπτυξη του μεταφραστή.

Αρχιτεκτονική

Η αρχιτεκτονική του μεταφραστή ως προς τα διακριτά πακέτα/εργαλεία παρουσιάζεται στην Εικόνα 3.1.

Εικόνα 3.1: Uml Διάγραμμα Πακέτων

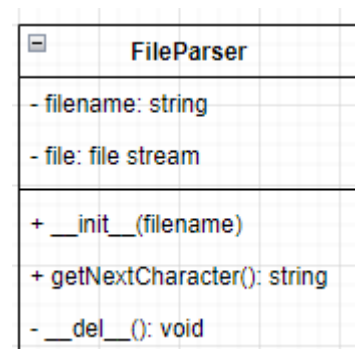


Το κάθε πακέτο περιγράφεται αναλυτικά στις επόμενες ενότητες.

Αναγνώστης Κειμένου

Το διάγραμμα κλάσεων για το πακέτο του αναγνώστη κειμένου παρουσιάζεται στην Εικόνα 3.2. Απαρτίζεται μόνο από την κλάση *FileParser*.

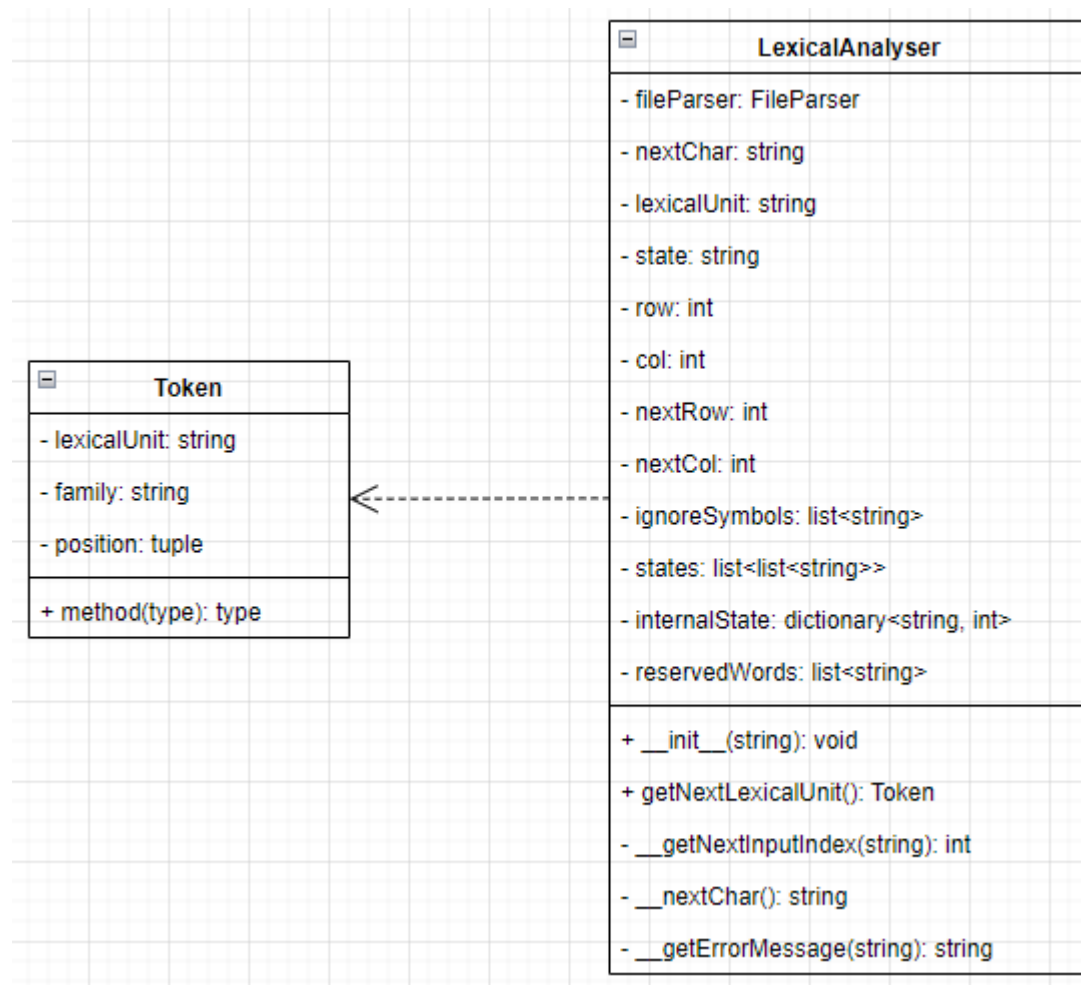
Εικόνα 3.2: Διάγραμμα κλάσεων για πακέτο αναγνώστη κειμένου



Λεκτικός Αναλυτής

Το διάγραμμα κλάσεων για το πακέτο του λεκτικού αναλυτή παρουσιάζεται στην Εικόνα 3.3.

Εικόνα 3.3: Διάγραμμα κλάσεων για πακέτο λεκτικού αναλυτή



Συντακτικός Αναλυτής

Το διάγραμμα κλάσεων για το πακέτο του συντακτικού αναλυτή παρουσιάζεται στην Εικόνα 3.4.

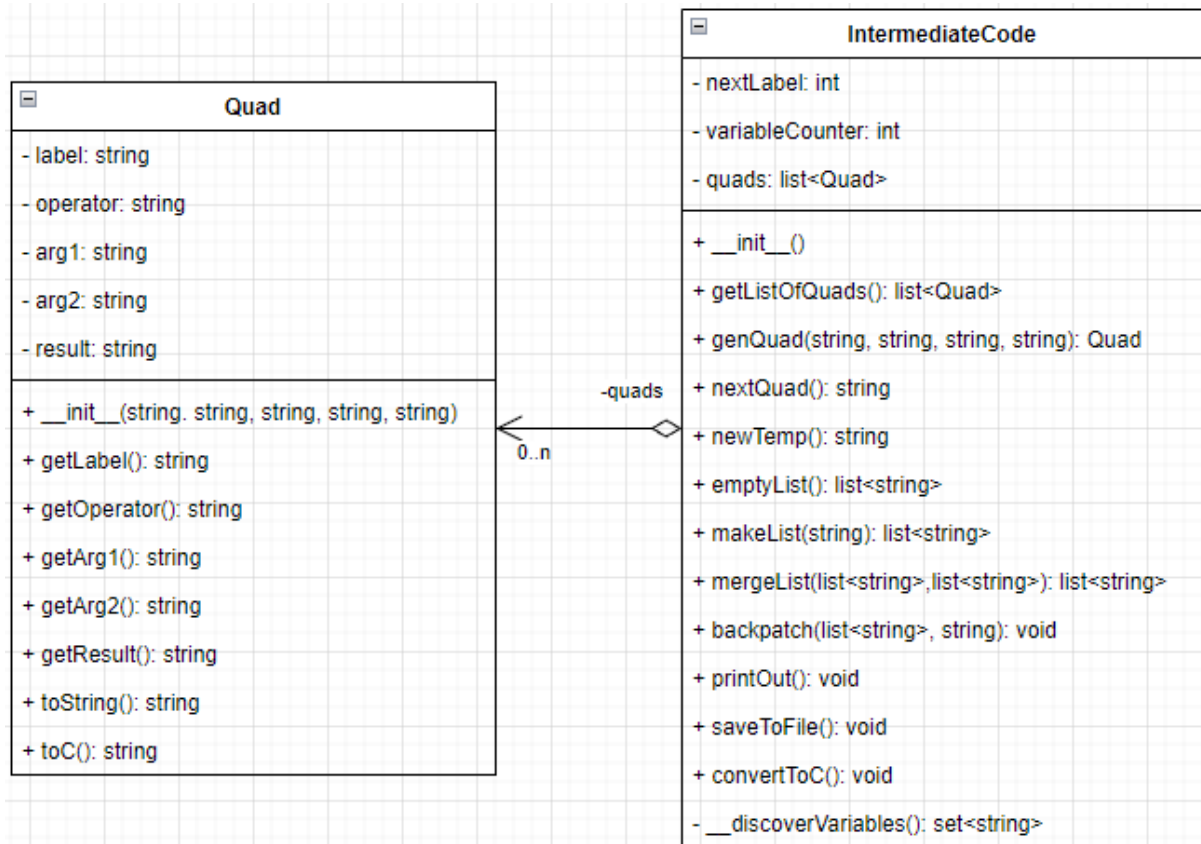
Εικόνα 3.4: Διάγραμμα κλάσεων για πακέτο συντακτικού αναλυτή



Ενδιάμεσος Κώδικας

Το διάγραμμα κλάσεων για το πακέτο του ενδιάμεσου κώδικα παρουσιάζεται στην Εικόνα 3.5.

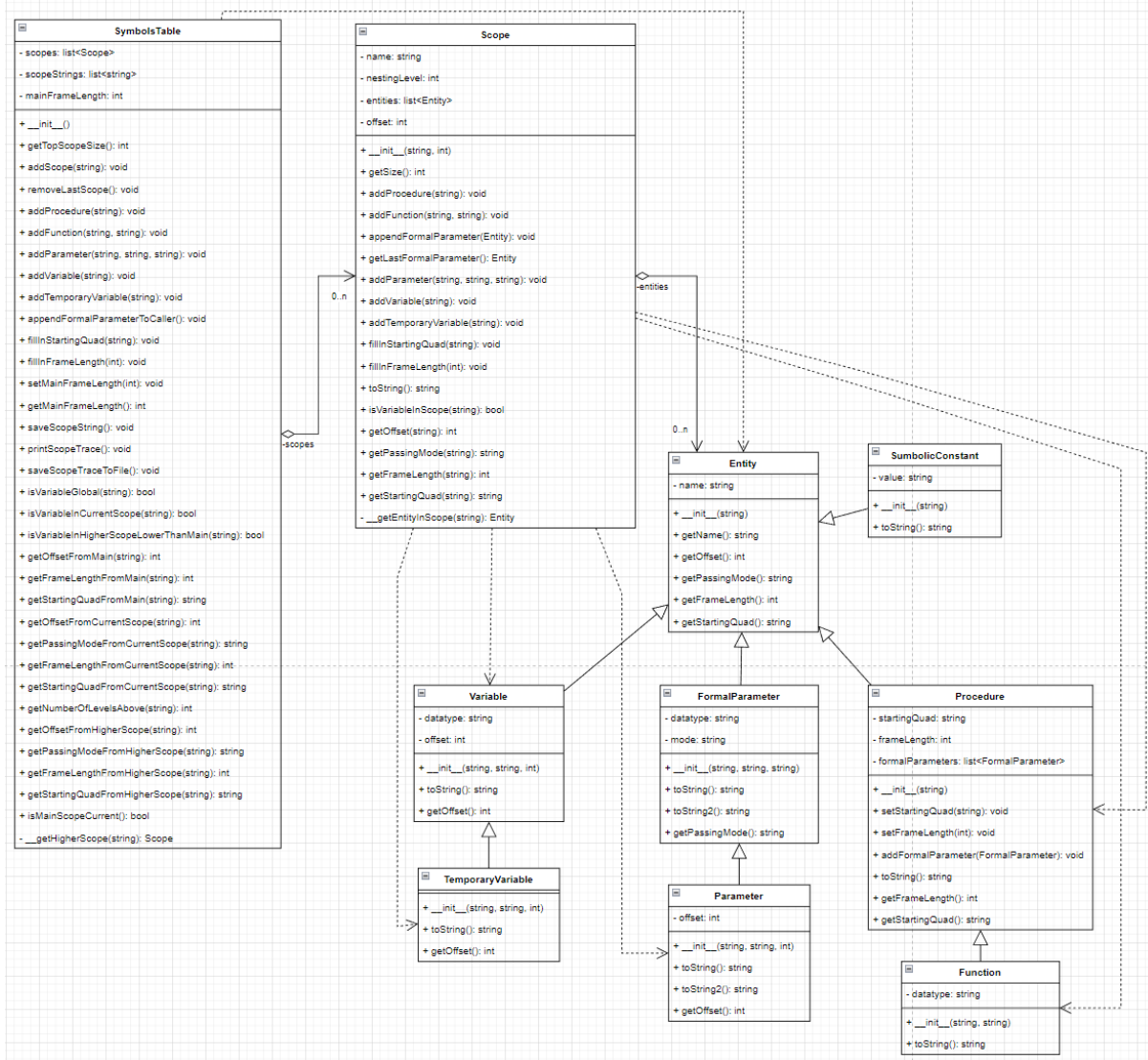
Εικόνα 3.5: Διάγραμμα κλάσεων για πακέτο ενδιάμεσου κώδικα



Πίνακας Συμβόλων

Το διάγραμμα κλάσεων για το πακέτο του πίνακα συμβόλων παρουσιάζεται στην Εικόνα 3.6.

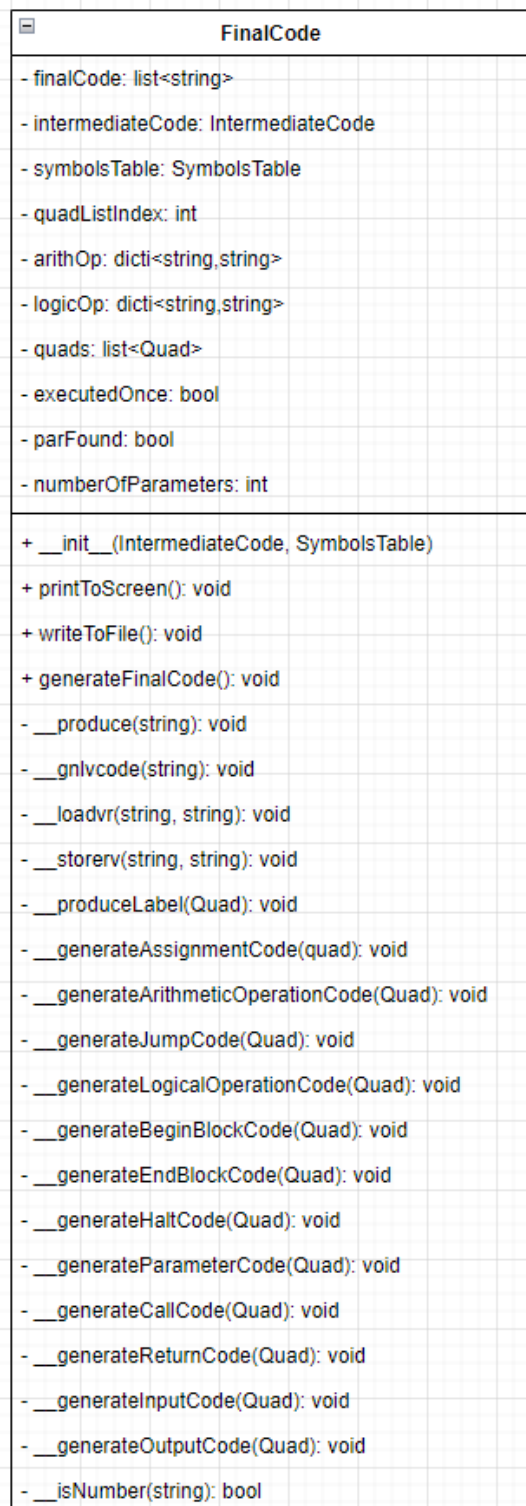
Εικόνα 3.6: Διάγραμμα κλάσεων για πακέτο πίνακα συμβόλων



Τελικός Κώδικας

Το διάγραμμα κλάσεων για το πακέτο του τελικού κώδικα παρουσιάζεται στην Εικόνα 3.7.

Εικόνα 3.7: Διάγραμμα κλάσεων για πακέτο τελικού κώδικα

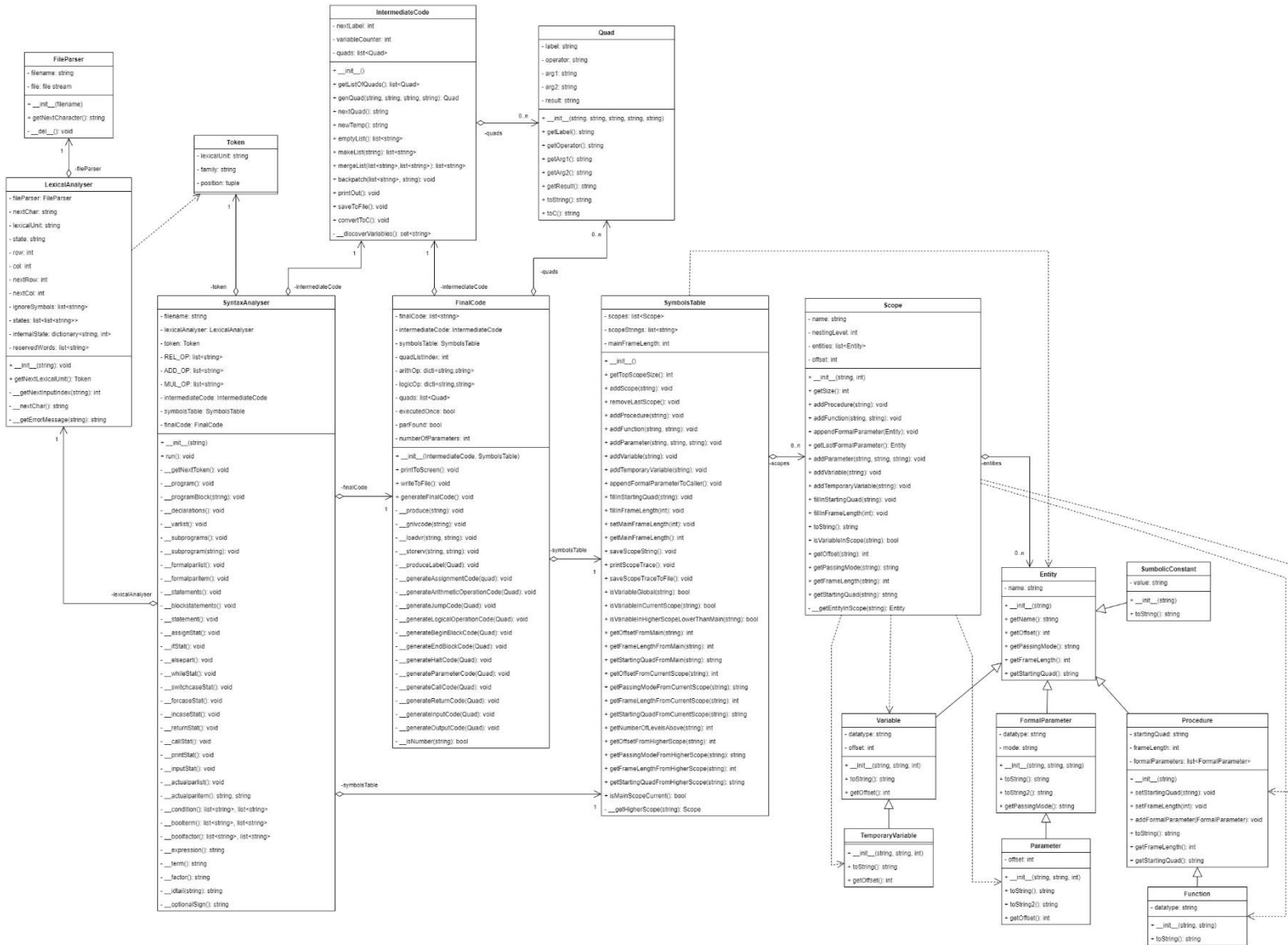


Συνολικά

Τέλος, η συνολική αρχιτεκτονική του μεταφραστή περιγράφεται από το συγκεντρωτικό διάγραμμα κλάσεων στην Εικόνα 3.8.

```

classDiagram
    class FileParser {
        -filename: string
        -file: file stream
        +__init__(filename)
        +getFileName(): string
        +getLine(): void
    }
    class Lexer {
        -lexer: FileParser
        -nextChar: string
        -lexicalUnit: string
        -state: string
        -row: int
        -col: int
        -nextRow: int
        -nextCol: int
        +ignoreSymbols: list-string
        +states: list-string
        +internalState: dictionary-string-int
        +reservedWords: list-string
        +__init__(string) void
        +getNextLexicalUnit() Token
        +getNextInputIndex(string) int
        +nextChar() string
        +getNextErrorMessage(string) string
    }
    class Token {
        -lexicalUnit: string
        -family: string
        -position: tuple
    }
    class SyntaxAnalyser {
        -filename: string
        -lexer: Lexer
        -token: Token
        -REL_OP: list-string
        -ADD_OP: list-string
        -MUL_OP: list-string
        -intermediateCode: IntermediateCode
        -symbolTable: SymbolsTable
        -finalCode: FinalCode
        +__init__(string) void
        +run() void
        +getNextToken() void
        +__program() void
        +__generateFinalCode() void
        +__declarations() void
        +__vars() void
        +__subprograms() void
        +__subprogram() void
        +__formalparams() void
        +__formalparam() void
        +__statements() void
        +__blockstatements() void
        +__statement() void
        +__assignment() void
        +__fstat() void
        +__elsepart() void
        +__whileStat() void
        +__switchCaseStat() void
        +__forCaseStat() void
        +__incDecStat() void
        +__returnStat() void
        +__callStat() void
        +__printStat() void
        +__inputStat() void
        +__actualparam() void
        +__condition() list-string
        +__boolean() list-string
        +__boolean() list-string
        +__boolean() list-string
        +__expression() string
        +__term() string
        +__factor() string
        +__dual() string
        +__optionalSign() string
    }
    class IntermediateCode {
        -nextLabel: int
        -variableCounter: int
        -quads: list-Quad
        +__init__()
        +getLetOfQuads() list-Quad
        +getQuad(string, string, string, string) Quad
        +newQuad() string
        +newTemp() string
        +emptyList() list-string
        +makeList(string) list-string
        +mergeList(list-string, list-string) list-string
        +backpatch(list-string, string) void
        +printOut() void
        +saveToFile() void
        +convertToC() void
        +discoverVariables() list-string
    }
    class Quad {
        -label: string
        -operator: string
        -arg1: string
        -arg2: string
        -result: string
        +__init__(string, string, string, string, string)
        +getLabel() string
        +getOperator() string
        +getArg1() string
        +getArg2() string
        +getResult() string
        +toString() string
        +toC() string
    }
    class SymbolsTable {
        -scopes: list-Scope
        -scopeStrings: list-string
        -mainFrameLength: int
        +__init__()
        +getTopScopeSize() int
        +addScope(string) void
        +removeLastScope() void
        +addProcedure(string) void
        +addFunction(string, string) void
        +addUnion(string, string) void
        +addParameter(string, string, string) void
        +addVariable(string) void
        +addTemporaryVariable(string) void
        +appendFormalParameterToCaller() void
        +findStartingQuad(string) void
        +findFrameLength(int) void
        +setMainFrameLength(int) void
        +setMainFrameLength() int
        +saveScope(string) void
        +printScopeTrace() void
        +saveScopeToCToFile() void
        +isVariableGlobal(string) bool
        +isVariableCurrentScope(string) bool
        +isVariableHigherScopeOverMainMain(string) bool
        +getOffsetFromMain(string) int
        +getFrameLengthFromMain(string) int
        +getStartingQuadFromMain(string) string
        +getOffsetFromCurrentScope(string) int
        +getPassingModeFromCurrentScope(string) string
        +getFrameLengthFromCurrentScope(string) int
        +getStartingQuadFromCurrentScope(string) string
        +getNumberOfLeavesAbove(string) int
        +getOffsetFromHigherScope(string) int
        +getPassingModeFromHigherScope(string) string
        +getFrameLengthFromHigherScope(string) int
        +getStartingQuadFromHigherScope(string) string
        +isMainScopeCurrent() bool
        +__getHigherScope(string) Scope
    }
    class Scope {
        -name: string
        -nestingLevel: int
        -entities: list-Entity
        -offset: int
        +__init__(string)
        +getScope() int
        +addProcedure(string) void
        +addFunction(string, string) void
        +appendFormalParameter(Entity) void
        +getFormalParameters() Entity
        +addParameter(string, string, string) void
        +addVariable(string) void
        +addTemporaryVariable(string) void
        +findStartingQuad(string) void
        +findFrameLength(int) void
        +toString() string
        +isVariableInScope(string) bool
        +getOffset(string) int
        +getPassingMode(string) string
        +getFrameLength(string) int
        +getStartingQuad(string) string
        +getEntityInScope(string) Entity
    }
    class Entity {
        -name: string
        +__init__(string)
        +getName() string
        +getOffset() int
        +getStartingQuad(string) string
        +getFrameLength() int
        +getStartingQuad() string
    }
    class SymbolicConstant {
        -value: string
        +__init__(string)
        +toString() string
    }
    class FormalParameter {
        -datatype: string
        -mode: string
        +__init__(string, string, string)
        +toString() string
        +toString2() string
        +getPassingMode() string
    }
    class Procedure {
        -startingQuad: string
        -frameLength: int
        -formalParameters: list-FormalParameter
        +__init__(string)
        +setStartingQuad(string) void
        +selfFrameLength(int) void
        +addFormalParameter(FormalParameter) void
        +toString() string
        +getFrameLength() int
        +getStartingQuad() string
    }
    class Variable {
        -datatype: string
        -offset: int
        +__init__(string, string, int)
        +toString() string
        +getOffset() int
    }
    class TemporaryVariable {
        +__init__(string, string, int)
        +toString() string
        +getOffset() int
    }
    class Parameter {
        -datatype: string
        -offset: int
        +__init__(string, string, int)
        +toString() string
        +toString2() string
        +getOffset() int
    }
    class Function {
        -datatype: string
        +__init__(string, string)
        +toString() string
    }
    FileParser --> Lexer
    Lexer --> SyntaxAnalyser
    SyntaxAnalyser --> Token
    SyntaxAnalyser --> IntermediateCode
    SyntaxAnalyser --> SymbolsTable
    SyntaxAnalyser --> Scope
    SyntaxAnalyser --> Entity
    SyntaxAnalyser --> SymbolicConstant
    SyntaxAnalyser --> FormalParameter
    SyntaxAnalyser --> Procedure
    SyntaxAnalyser --> Variable
    SyntaxAnalyser --> TemporaryVariable
    SyntaxAnalyser --> Parameter
    SyntaxAnalyser --> Function
    IntermediateCode --> Quad
    SymbolsTable --> Scope
    SymbolsTable --> Entity
    SymbolsTable --> SymbolicConstant
    SymbolsTable --> FormalParameter
    SymbolsTable --> Procedure
    SymbolsTable --> Variable
    SymbolsTable --> TemporaryVariable
    SymbolsTable --> Parameter
    SymbolsTable --> Function
    Scope --> Entity
    Scope --> SymbolicConstant
    Scope --> FormalParameter
    Scope --> Procedure
    Scope --> Variable
    Scope --> TemporaryVariable
    Scope --> Parameter
    Scope --> Function
    Entity --> SymbolicConstant
    Entity --> FormalParameter
    Entity --> Procedure
    Entity --> Variable
    Entity --> TemporaryVariable
    Entity --> Parameter
    Entity --> Function
    SymbolicConstant --> FormalParameter
    SymbolicConstant --> Procedure
    SymbolicConstant --> Variable
    SymbolicConstant --> TemporaryVariable
    SymbolicConstant --> Parameter
    SymbolicConstant --> Function
    FormalParameter --> Procedure
    FormalParameter --> Variable
    FormalParameter --> TemporaryVariable
    FormalParameter --> Parameter
    FormalParameter --> Function
    Procedure --> Variable
    Procedure --> TemporaryVariable
    Procedure --> Parameter
    Procedure --> Function
    Variable --> TemporaryVariable
    Variable --> Parameter
    Variable --> Function
    TemporaryVariable --> Parameter
    TemporaryVariable --> Function
    Parameter --> Function
  
```



CRC Cards

Οι Python κλάσεις του μεταφραστή περιγράφονται με περισσότερη λεπτομέρεια στις παρακάτω ενότητες για κάθε πακέτο της αρχιτεκτονικής.

Αναγνώστης Κειμένου

Οι κλάση του πακέτου αναγνώστη κειμένου περιγράφεται στον Πίνακας 3.1.

Πίνακας 3.1: CRC card για την κλάση *FileParser*

Όνομα κλάσης: FileParser	
Αρμοδιότητα: <ul style="list-style-type: none">▪ Άνοιγμα αρχείου πηγαίου κώδικα.▪ Ανάγνωση αρχείου χαρακτήρα-χαρακτήρα.	Εξαρτήσεις: <ul style="list-style-type: none">▪ καμία

Λεκτικός Αναλυτής

Οι κλάσεις του πακέτου λεκτικός αναλυτής καταγράφονται στους: Πίνακας 3.2 και Πίνακας 3.3.

Πίνακας 3.2: CRC card για την κλάση *Token*

Όνομα κλάσης: Token	
Αρμοδιότητα: <ul style="list-style-type: none">▪ Δομή αποθήκευσης πληροφορίας για κάθε λεκτική μονάδα.	Εξαρτήσεις: <ul style="list-style-type: none">▪ καμία

Πίνακας 3.3: CRC card για την κλάση *LexicalAnalyser*

Όνομα κλάσης: LexicalAnalyser	
Αρμοδιότητα: <ul style="list-style-type: none">▪ Χρησιμοποιεί τον αναγνώστη κειμένου για να προσπελάσει το αρχείο πηγαίου κώδικα.▪ Αναγνωρίζει και επιστρέφει την κάθε λεκτική μονάδα.▪ Παράγει κατάλληλα πληροφοριακά μηνύματα και σταματάει την εκτέλεση όταν ανακαλύψει λεκτική μονάδα που δεν πληροί τους κανόνες της γλώσσας <i>cimble</i>.	Εξαρτήσεις: <ul style="list-style-type: none">▪ FileParser▪ Token

Συντακτικός Αναλυτής

Οι κλάση του πακέτου συντακτικός αναλυτής παρουσιάζεται στον Πίνακα 3.4.

Πίνακας 3.4: CRC card για την κλάση *SyntaxAnalyser*

Όνομα κλάσης: <i>SyntaxAnalyser</i>	
Αρμοδιότητα: <ul style="list-style-type: none">▪ Ελέγχει τον πηγαίο κώδικα για κάθε λεκτική μονάδα την φορά για να διαπιστώσει αν το πρόγραμμα αντιστοιχεί στην γλώσσα <i>cimple</i>.▪ Καλεί μεθόδους της κλάσης του ενδιάμεσου κώδικα σε κατάλληλα σημεία ώστε να δημιουργήσει τις τετράδες του ενδιάμεσου κώδικα.▪ Καλεί μεθόδους της κλάσης του πίνακα συμβόλων σε κατάλληλα σημεία ώστε να δημιουργήσει και να κρατά ενημερωμένο τον πίνακα συμβόλων ανάλογα με το σημείο της μετάφρασης.▪ Καλεί μεθόδους της κλάσης του τελικού κώδικα σε κατάλληλα σημεία ώστε να δημιουργήσει τον τελικό κώδικα.	Εξαρτήσεις: <ul style="list-style-type: none">▪ <i>LexicalAnalyser</i>▪ <i>IntermediateCode</i>▪ <i>SymbolsTable</i>▪ <i>FinalCode</i>

Ενδιάμεσος Κώδικας

Οι κλάσεις του πακέτου ενδιάμεσου κώδικα καταγράφονται στους: Πίνακας 3.5 και Πίνακας 3.6.

Πίνακας 3.5: CRC card για την κλάση *Quad*

Όνομα κλάσης: <i>Quad</i>	
Αρμοδιότητα: <ul style="list-style-type: none">▪ Δομή αποθήκευσης πληροφορίας για κάθε τετράδα του ενδιάμεσου κώδικα.	Εξαρτήσεις: <ul style="list-style-type: none">▪ καμία

Πίνακας 3.6: CRC card για την κλάση *IntermediateCode*

Όνομα κλάσης: IntermediateCode	
Αρμοδιότητα: <ul style="list-style-type: none"> ▪ Παράγει τις τετράδες του ενδιάμεσου κώδικα παράλληλα με την συντακτική ανάλυση. ▪ Παρέχει μεθόδους για την εμφάνιση στην οθόνη αλλά και αποθήκευση στον δίσκο του τελικού κώδικα. ▪ Παρέχει μέθοδο για την μετατροπή του συνόλου των τετράδων σε ισοδύναμο πρόγραμμα c. 	Εξαρτήσεις: <ul style="list-style-type: none"> ▪ Quad

Πίνακας Συμβόλων

Οι κλάσεις του πακέτου πίνακα συμβόλων καταγράφονται στους: Πίνακας 3.7 έως Πίνακας 3.16.

Πίνακας 3.7: CRC card για την κλάση *SymbolsTable*.

Όνομα κλάσης: SymbolsTable	
Αρμοδιότητα: <ul style="list-style-type: none"> ▪ Δημιουργεί τον πίνακα συμβόλων υπό την μορφή μιας συλλογής περιοχών (scopes). ▪ Κρατάει τον πίνακα συμβόλων ενημερωμένο ανάλογα με το σημείο της μετάφρασης του προγράμματος <i>cimple</i>. ▪ Παρέχει διεπαφή (API) για τις κλάσεις του συντακτικού αναλυτή και τελικού κώδικα ώστε να πραγματοποιείται επικοινωνία με τις οντότητες του πίνακα συμβόλων. ▪ Παρέχει μεθόδους για την εμφάνιση στην οθόνη αλλά και αποθήκευση στον δίσκο του τελικού κώδικα. 	Εξαρτήσεις: <ul style="list-style-type: none"> ▪ Scope ▪ Entity

Πίνακας 3.8: CRC card για την κλάση Scope.

Όνομα κλάσης: Scope	
Αρμοδιότητα: <ul style="list-style-type: none"> Αποθηκεύει τις υπάρχουσες οντότητες για κάθε περιοχή (scope) του υπό μετάφραση προγράμματος. Παρέχει διεπαφή (API) για την επικοινωνία με τις οντότητες. 	Εξαρτήσεις: <ul style="list-style-type: none"> Entity Variable TemporaryVariable FormalParameter Parameter Procedure Function SymbolicConstant

Πίνακας 3.9: CRC card για την κλάση Entity.

Όνομα κλάσης: Entity	
Αρμοδιότητα: <ul style="list-style-type: none"> Γενική (abstract) δομή αποθήκευσης πληροφορίας για κάθε οντότητα του πίνακα συμβόλων. Αποτελεί την κορυφή στο δέντρο της ιεραρχίας των κλάσεων των υπολοίπων οντοτήτων. 	Εξαρτήσεις: <ul style="list-style-type: none"> καμία

Πίνακας 3.10: CRC card για την κλάση Variable.

Όνομα κλάσης: Variable	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε μεταβλητή του προγράμματος <i>cimble</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> Entity

Πίνακας 3.11: CRC card για την κλάση TemporaryVariable.

Όνομα κλάσης: TemporaryVariable	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε προσωρινή μεταβλητή του προγράμματος <i>cimble</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> Variable

Πίνακας 3.12: CRC card για την κλάση *FormalParameter*.

Όνομα κλάσης: FormalParameter	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε παράμετρο (κατά την δήλωση συνάρτησης) του προγράμματος <i>cimprle</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> Entity

Πίνακας 3.13: CRC card για την κλάση *Parameter*.

Όνομα κλάσης: Parameter	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε παράμετρο (κατά την κλήση συνάρτησης) του προγράμματος <i>cimprle</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> FormalParameter

Πίνακας 3.14: CRC card για την κλάση *Procedure*.

Όνομα κλάσης: Procedure	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε διαδικασία του προγράμματος <i>cimprle</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> Entity

Πίνακας 3.15: CRC card για την κλάση *Function*.

Όνομα κλάσης: Function	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε συνάρτηση του προγράμματος <i>cimprle</i> που εισάγεται στον πίνακα συμβόλων. 	Εξαρτήσεις: <ul style="list-style-type: none"> Procedure

Πίνακας 3.16: CRC card για την κλάση *SymbolicConstant*.

Όνομα κλάσης: SymbolicConstant	
Αρμοδιότητα: <ul style="list-style-type: none"> Δομή αποθήκευσης πληροφορίας για κάθε σταθερά του προγράμματος <i>cimprle</i> που εισάγεται στον πίνακα συμβόλων. (Στην <i>cimprle</i> δεν υπάρχουν συμβολικές σταθερές) 	Εξαρτήσεις: <ul style="list-style-type: none"> Entity

Τελικός Κώδικας

Οι κλάση του πακέτου τελικού κώδικα περιγράφεται στον Πίνακας 3.17.

Πίνακας 3.17: CRC card για την κλάση *FinalCode*.

Όνομα κλάσης: FinalCode	
Αρμοδιότητα: <ul style="list-style-type: none"> Παράγει εντολές τελικού κώδικα για μια σειρά από τετράδες ενδιαμέσου κώδικα που αντιστοιχούν στην πλήρη μετάφραση μιας περιοχής (scope). Χρησιμοποιεί τον πίνακα συμβόλων για να αντλεί πληροφορίες για τις οντότητες και τις περιοχές (scopes) που αυτές ανήκουν. Παρέχει μεθόδους για την εμφάνιση στην οθόνη αλλά και αποθήκευση στον δίσκο του τελικού κώδικα. 	Εξαρτήσεις: <ul style="list-style-type: none"> IntermediateCode SymbolsTable Quad