**Компајлер за Микро Јаву**

Програмски преводиоци 1

Професор: Драган Бојић

Асистенти: Маја Вукасовић,  
Михајло Огризовић

Студент: Урош Стајић  
2018/0073

Садржај

[Кратак опис пројекта 3](#_Toc61829248)

[Опис команди: 3](#_Toc61829249)

[Кратак опис тестова 4](#_Toc61829250)

[Кратак опис класа 5](#_Toc61829251)

## **Кратак опис пројекта**

Циљ пројектног задатка је рализација компајлера за програмски језик Микројаву. Сам пројекат се састоји из четири фаза:

-лексичка анализа  
 -синтаксна анализа  
 -семантичка анализа  
 -генерисање кода

Лексичка анализа се базира на препознавању језичких лексема и враћању скупа токена издвојених из изворног кода. У ту намену коришћен је алат Jflex.

Синтаксна анализа има улогу да утврди да ли добијени токени из претходне фазе могу да формирају граматички исправне секвенце. Приликом синтаксне анализе коришћен је алат CUP за генерисање парсера. Као крајњи производ синтаксне аналите, генерисано је апстрактно синтаксно стабло, на основу граматике написане у овој фази.

Семантичка анализа креће од обиласка апстрактног синтаксног стабла добијеног у претходној фази. Имплементирају се методе за посећивање чворова у којих се испитује исправност одређених семантичких правила. На основу ове фазе се попуњава и табела симбола.

Генератор кода сада користи све продукте претходног рада како превео синтаксно и семантички исправан програм у извршни облик. Овде су се такође правиле методе за посећивање чворова. За писање асемблерског кода коришћена је класа Code.

## 

## **Опис команди**:

1. На почетку је битно генерисати класу Yylex.java која представља лексер, она се бави препознавањем језичких лексема и враћању скупа токена и у ову сврху је направљен фајл lexer.flex. Када се покрене таргет LexerGen, уколико је све исправно, генерисаће се ова класа.
2. Када се успешно генерише Yylex.java, следећи корак је генерисање MJParser.java тј. генерисање парсера. У овој ситацији користи се таргет “Parser Gen” .
3. Онда треба и позвати таргет “repackage” како би се сви делови спаковали у одговарајуће фолдере.
4. И коначно, покреће се "compile" како би се добио exe фајл.
5. Онда је могуће покренути и програм уз помоћ класе Run. Покреће се таргер “runObj”.

Како би се омогућило све наведено, користе се следећи jar фајлови:

mj-runtime-1.1.jar (одавде нам је битна Рун класа)  
 symboltable-1-1.јаr (табела симбола се користи из ове класе)  
 cup\_v10k.jar  
 Jflex.jar  
 Log4j-1.2.17.jar (користи се за лакши испис)

## **Кратак опис тестова**

У оквиру фолдера тестови се налазе примери тестова.

- Тест1 садржи опоравак од грешке.

- Тест2 садржи доста инструкција обухваћених граматиком као што су, коришћење елемената низа, лабеле, аритметичке операције...

-Тест3 садржи неке функционалности везане за ниво Б, као што су if/else, do while...

- Тест301\_А представља јавни тест за ниво А.

- Тест301 представља јавни тест за ниво А, уколико се ради пројекат за Б или Ц.

-Тест302 представља јавни тест за ниво Б

## 

## **Кратак опис новоиведених класа**

CodeGenerator.java

Класа која служи за генерисање кода програма који је прошао синтаксну и семантичку проверу. Кроз visit методе се каже одређеној класи шта треба да ради у којој ситуацији.

Compiler.java

Класа која је задужена за компајлирање програма. Прво проверава да ли је одговарајући број улазних параметара. Затим позива парсер како би извршио лексичку и синтаксну анализу нашег програма. Уколико је улазни програм прошао синтаксну анализу компајлер позива семантичку анализу помоћу класе SemanticAnalyzator.java. Ако је програм прошао семантичку анализу онда компајлер позива генерисање кода помоћу класе CodeGenerator.java. Приликом генерисања кода прави се фајл који садржи објектни код који ће се извршавати у виртуелној машини за Микро Јаву

MojaTabela.java

Класа која наслеђује Tab.java. Почетна табела симбола Tab.java садржи све битне функционалности сем bool типа, па у ту сврху је и направљена ова класа, где је додат тип.

MyDumpSymbolTableVisitor.java

Служи као помоћ при имплементирању методе void tsdump() која служи за испис табеле симбола. Пошто класа наслеђује класу SymbolTableVisitor, у њој су редефинисане захтеване методе,а све у циљу прегледнијег исписа табеле симбола.

SemanticAnalyzer.java

У овој класи су имплементирана сва ограничења наше граматике кроз visit методе. Служи да провери да ли је написан код семантички исправан и у зависности од тога јави поруку да ли је парсирање успешно или неуспешно.