|  |  |
| --- | --- |
| etf_logo_2010_cir | **УНИВЕРЗИТЕТ У БЕОГРАДУ**  **ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ**  Програмски преводиоци 1 – 2021/2022.  Булевар краља Александра 73, ПФ 35-54, 11120 Београд, Србија  телефон: 011/3218-321, [dekanat@etf.bg.ac.rs](mailto:dekanat@etf.bg.ac.rs) |

ПРОЈЕКТНИ ЗАДАТАК

***КОМПАЈЛЕР ЗА МИКРОЈАВУ***

Програмски преводиоци 1

Професор: Драган Бојић

Асистенти: Маја Вукасовић,

Михајло Огризовић

Студент: Милош Стојановић, 2018/0228

Београд, август 2022.

Садржај

[**1.** **Опис пројектног задатка** 1](#_Toc111676069)

[1.1. Компајлер 1](#_Toc111676070)

[1.2. Лексички анализатор 1](#_Toc111676071)

[1.3. Синтаксни анализатор 1](#_Toc111676072)

[1.4. Семантички анализатор 1](#_Toc111676073)

[1.5. Генератор кода 2](#_Toc111676074)

[**2.** **Покретање компајлера** 2](#_Toc111676075)

[**3.** **Тестирање** 2](#_Toc111676076)

[**4.** **Новоуведене класе** 3](#_Toc111676077)

# **Опис пројектног задатка**

Пројектни задатак је писан у програмском језику **Java** користећи **JavaSE1.8 JRE**. За израду задатка је коришћено развојно окружење **Eclipse 2021-12**. Додатно, пројекат користи јавно доступне библиотеке **JFlex.jar**, **cup\_v10k.jar** и **log4j-1.2.17.jar** као и библиотеке дате уз пројектни задатак **mj-runtime-1.1.jar** и **symboltable-1.1.jar**.

Циљ пројектног задатка је **рализација компајлера за програмски језик Микројаву.**

## Компајлер

**Компајлер** пружа могућност превођења синтаксно и семантички исправних Микројава програма у Микројава бајткод који се извршава на Микројава виртуелној машини, уз способност опоравка од грешке за предвиђене ситуације. Компајлер садржи **четири** основне функционалности: лексичку анализу, синтаксну анализу, семантичку анализу и генерисање кода.

## Лексички анализатор

**Лексички анализатор** пружа могућност препознавања језичких лексема и на крају враћа скуп токена издвојених из изворног кода, који се даље разматрају у оквиру синтаксне анализе. Лексички анализатор је имплементиран користећи **JFlex.jar** библиотеку. У случају да лексички анализатор не може да очита неки токен, пријављује лексичку грешку као и линију на којој се догодила.

## Синтаксни анализатор

**Синтаксни анализатор** има задатак да утврди да ли претходно добијени токени могу да формирају граматички исправне реченице које испуњавају правила пројектног задатка. Као резултат генерише апстаракно синтаксно стабло, на основу дате граматике. Синтаксни анализатор је имплементиран користећи **cup\_v10k.jar** библиотеку. За неке предвиђене ситуације грешака у улазном програму, синтаксни анализатор је способан да изврши опоравак од грешке док, ако се деси непредвиђена грешка, пријављује фаталну грешку и зауставља даљу обраду.

## Семантички анализатор

**Семантички анализатор** обилази добијено апстарактно синтаксно стабло и посећује чворове користећи bottom-up парсирање. Приликом посете чворова, проверава поштовање семантичких правила датих у пројектном задатку а у случају непоштовања неког правила зауставља даљу обраду и генерише грешку са једнозначним описом проблема. На крају свог извршавања, приказује стање генерисане табеле симбола. За његову израду се користи **symboltable-1.1.jar**.

## Генератор кода

**Генератор кода** преводи синтаксно и семантички исправне програме у извршни облик за Микројава виртуелну машину. У случају неке грешке у улазном коду, генерисаће се runtime грешка приликом извршавања на виртуелној машини. За његову израду се користи **mj-runtime-1.1.jar**.

# **Покретање компајлера**

За извршавање кода из *program.mj* фајла користи се класа „*Run*“ из библиотеке **mj-runtime-1.1.jar**. Покретање се врши помоћу *RunObj* опције у Ant фајлу **build.xml** која ће покренути секвенцијално прво класу „*disasm*“ за дисасемблирање кода, затим „*DebugObj*“ за покретање кода у режиму дебаговања и на крају „*RunObj*“ чиме се поново покреће код али у нормалном режиму да би се прегледно видео испис на крају извршавања. *RunObj* као параметар прима *program.obj* фајл који представља компајлован *program.mj*.

Да би се генерисао потребан *program.obj* фајл додатно је уведена класа **Compiler.java**.

Пре покретања наведене класе неоходно је покренути *Compile* опцију у поменутом **build.xml** фајлу чиме ће се изгенерисати неопходне јава класе, након чега се може покренути процес компајлирања.

За читљивији испис је задужена још и библиотека **log4j.jar**.

# **Тестирање**

У оквиру фолдера тестови се налазе примери тестова.

* **test301\_avg** представља јавни тест за ниво А.
* **test302\_avg** представља јавни тест за ниво Б.
* **2\_myTest** садржи репрезентативни скуп инструкција за тестирање исправности синтаксног анализатора.
* **2\_testErrorRecovery** садржи цео скуп грешака за које је предвиђен опоравак од грешке у фази синтаксне анализе.
* **3\_myFinalTest** садржи репрезентативни скуп инструкција за тестирање исправности семантичког анализатора.
* **3\_test\_error\_semantics** садржи скуп инструкција које приказују прецизност детектовања семантичких грешака у улазном фајлу.

# 

# **Новоуведене класе**

У оквиру фолдера са изворним кодом налазе се и класе, поред класа које представљају имплементацију решења и класа које су аутоматски генерисане радом спецификације, које су помоћног карактера и служе да олакшају реализацију коначног решења.

* **CounterVisitor** служи да изброји колико параметара има у методи.
* **DoWhileAdrHelper** служи као помоћ при разрешавању адреса скокова при проверавању испуњености услова у do-while петљи.
* **ElseThenAdrHelper** служи као помоћ при разрешавању адреса скокова при проверавању испуњености услова у if-else гранањима.
* **MyLabel** служи као помоћ при разрешавању адреса скокова на лабеле.