**Projekat iz programskih prevodilaca I Kompajler za MikroJavu**

*Dokumentacija*

Vuković Sara

2018/430

# **Uvod**

U okviru ovog projekta se razvija kompajler za jezik *MicroJava.* MicroJava je jezik veoma sličan Javi, ali sa mnogo manjim opsegom funkcionalnosti.

Razvijanje kompajlera se obavlja u četiri faze:

* Leksička analiza
* Sintaksna analiza
* Semantička analiza
* Generisanje koda

U okviru prve faze se piše .flex specifikacija i time se formira leksički analizator. On skenira .mj fajl i detektuje leksičke strukture definisane u .flex fajlu.

U okviru druge faze se piše .cup specifikacija. U njoj se definiše LALR(1) gramatika na osnovu specifikacije MikroJava jezika. Time se formira sintaksni analizator iliti parser. U toku pisanja specifikacije, svakoj produkciji se dodaje naziv, pomocu kojeg AST-CUP alat generiše apstraktno sintaksno stablo. U okviru ove faze se takođe implementira oporavak od sintaksnih grešaka.

U trećoj fazi se implementira SemanticAnalzyer klasa, koja proširuje automatski generisanu klasu *VisitorAdapter.* U njoj se redefinišu visit metode čvorova relevantnih za semantičku analizu. Unutar tih metoda se vrši provera konteksnih uslova opisanih u specifikaciji MikroJave. U okviru ove faze se ažurira i tabela simbola.

Četvrta faza obuhvata generisanje bajtkoda izvršivog na *MikroJava virtuelnoj mašini.* Izlaz generatora koda je izvršiv .obj fajl.

# **Pokretanje i testiranje rešenja**

Projekat se pokreće uz pomoć build.xml skripte u kojoj je opisano kojim redosledom i koje komande se pozivaju, kako bi se na kraju dobio funkcionalan kompajler i generator bajtkoda za MikroJavu.

Prvo se generiše lekser, na osnovu .flex fajla, upotrebom sledećeg dela skripte:

A computer code with text

Description automatically generated

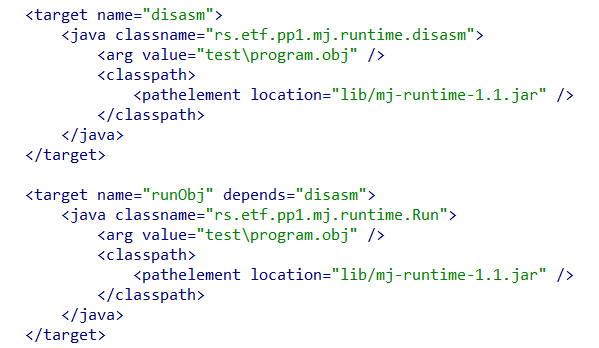
Zatim se generiše parser uz pomoć JCup alata, a na osnovu napisane .cup specifikacije:

A screenshot of a computer code

Description automatically generated

Nakon toga, uz pomoć SemanticAnalyzer klase se obavlja obilazak prethodno generisanog sintaksnog stabla i vrši se provera semantičkih pravila.

Ukoliko su sve prethodne provere prošle, uz pomoć CodeGenerator klase se generiše kod, na osnovu kojeg se dobija .obj fajl, koji može da se pokrene u okviru MikroJava virtuelne mašine uz pomoć Run klase, iz mj-runtime bibilioteke.  
  
Takođe, dobijeni .obj fajl može i da se disasemblira, uz pomoć disasm klase iz gore pomenute biblioteke.



# **Priloženi test primeri**

U okviru test foldera se nalazi 7 .mj test fajlova:

* sintaksnaAnalizaGreske.mj – u okviru ovog fajla imamo sintaksne greske od kojih parser ne moze da se oporavi
* sintaksnaAnalizaOporavakOdGreske.mj – u okviru ovog fajla se prikazuju sve greske od kojih parser moze da se oporavi
* sintaksnaAnaliza.mj – jedan pravilno napisan MikroJava kod
* semantickaAnalizaSaGreskama.mj – test u kojem se nalaze razne semanticki neispravne instrukcije
* semantickaAnaliza.mj – sintaksno i semanticki ispravan kod
* test301.mj – javni test za projekat koji je rađen za nivo A
* test302.mj – javni test za projekat koji je rađen za nivo B

# **Novouvedene klase**

Novouvedena klasa ***StructExt*** je klasa koja proširuje ***Struct*** klasu iz symboltable biblioteke. Proširenje se ogleda u tome što je uvedena nova, statička metoda, koja kao parametre prima dva objekta, Struct tipa, i proverava da li su oni kompatibilni pri dodeli vrednosti. Po specifikaciji MikroJave, dva tipa su kompatibilna za dodelu vrednosti ako su ekvivalentni ili ukoliko je tip destinacione promenljive referenca, a tip izvorne promenljive je null