**PROJEKAT IZ PROGRAMSKIH PREVODILACA 1**

-Microjava kompajler-

Student: Miloš Kovačević

Broj indeksa: 0385/2019

# Opis postavke zadatka

Cilj projektnog zadatka je realizacija kompajlera za programski jezik Mikrojavu. Kompajler omogućava prevodjenje sintaksno i semantički ispravnih Mikrojava programa u Mikrojava bajtkod koji se izvršava na virtuelnoj mašini za Mikrojavu. Sintaksno i semantički ispravni Mikrojava programi su definisani specifikacijom.

Programski prevodilac za Mikrojavu ima četiri osnovne funkcionalnosti: leksičku analizu, sintaksnu analizu, semantičku analizu i generisanje koda.

*Leksički analizator* treba da prepoznaje jezičke lekseme i vrati skup tokena izdvojenih iz izvornog koda, koji se dalje razmatraju u okviru sintaksne analize. Ukoliko se tokom leksičke analize detektuje leksička greška, potrebno je ispisati odgovarajuću poruku na izlaz.

*Sintaksni analizator* ima zadatak da utvrdi da li izdvojeni tokeni iz izvornog koda programa mogu formiraju gramatički ispravne sentence. Tokom parsiranja Mikrojava programa potrebno je na odgovarajući način omogućiti i praćenje samog procesa parsiranja na način koji će biti u nastavku dokumenta detaljno opisan. Nakon parsiranja sintaksno ispravnih Mikrojava programa potrebno je obavestiti korisnika o uspešnosti parsiranja. Ukoliko izvorni kod ima sintaksne greške, potrebno je izdati adekvatno objašnjenje o detektovanoj sintaksnoj grešci, izvršiti oporavak i nastaviti parsiranje.

*Semantički analizator* se formira na osnovu apstraktnog sintaksnog stabla koje je nastalo kao rezultat sintaksne analize. Semantička analiza se sprovodi implementacijom metoda za posećivanje čvorova apstraktnog sintaksnog stabla. Stablo je formirano na osnovu gramatike implementirane u prethodnoj fazi. Ukoliko izvorni kod ima semantičke greške, potrebno je prikazati adekvatnu poruku o detektovanoj semantičkoj grešci.

*Generator koda* prevodi sintaksno i semantički ispravne programe u izvršni oblik za odabrano izvršno okruženje Mikrojava VM. Generisanje koda se implementira na sličan način kao i semantička analiza, implementacijom metoda koje posećuju čvorove.

# Pokretanje rješenja

Za početak je neophodno izgenerisati lekser i parser, a to se radi uz pomoć Ant alata. U korjenom direktorijumu projekta postoji build.xml skripta koju koristi Ant alat i nad njom se isti treba pokrenuti tako što se izabere meta *compile.*

Nakon toga sve što je potrebno uraditi je iskoristiti postojeću ili napraviti novu Run konfiguraciju nad klasom test/rs.ac.bg.etf.pp1.Compiler, a kao argumente komandne linije postaviti putanju ka izvornom Microjava kodu, putanju ka izlaznom .obj fajlu i (opciono) naznake za preusmjeravanje standardnog izlaza i izlaza za greške, u tom redoslijedu.

# Opis napisanih testova

Testovi su napisani u svrhu testiranja svih elementat rješenja. Među njima ima sintaksno i semantičko ispravno i neispravno napisanih programa koji provjeravaju kontekstne uslovne i semantičke uslove. Za sve neipravno napisane programe rješenje ispisuje odgovarajuće poruke i greške, čijim tumačenjem se lako može lokalizovati i otkloniti posmatrana greška.

# Novouvedene klase

Pored automatski generisanih i klasa koje su po postavci obavezan dio rješenja, uvedene su i sljedeće dvije klase: CounterVisitor i MyTab.

CounterVisitor predstavlja pomoćnu klasu koja proširuje automatski generisani klasu VisitorAdaptor i služi kao osnovna klasa za dvije unutrašnje klase čija je jedina svrha da izbroje formalne parametre i broj deklarisanih lokalnih promjenljivih kod metoda.

MyTab proširuje Tab klasu iz biblioteke *symboltable-1-1.jar* u svrhe dodavanja strukturnog čvora za promjenljive tima bool, kao i strukturne čvorove za nizove svih primitivnih tipova, ali se ti ne koriste u rješenju.