Univerzitet u Beogradu, Elektrotehnički fakultet

**MicroJava kompajler**

Projekat iz predmeta Programski prevodioci 1

Student:

Aleksandar Dinčić 2018/0028

# Opis projekta

Cilj projektnog zadatka je realizacija **kompajlera** za programski jezik **Mikrojava**. Kompajler omogućava prevođenje sintaksno i semantički ispravnih Mikrojava programa u Mikrojava bajtkod koji se izvršava na virtuelnoj mašini za Mikrojavu. Prevodilac ima četiri osnovne funkcionalnosti: leksičku analizu, sintaksnu analizu, semantičku analizu i generisanje koda.

**Leksički analizator** 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, ispisuje odgovarajuću poruku na izlaz.

**Sintaksni analizator** utvrđuje da li izdvojeni tokeni iz izvornog koda programa mogu da formiraju gramatički ispravne sentence. Ukoliko izvorni kod ima sintaksne greške izaje se adekvatno objašnjenje o detektovanoj sintaksnoj grešci, izvršava oporavak i nastavlja 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, prikazuje se adekvatna poruka 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.

# Opis komandi

Željenu operaciju korisnik pokreće koriščenjem komandi iz priložene Apache Ant skripte, kao i pokretanjem odgovarajućih Java klasa.

**Generisanje klasa leksera** vrši se pokretanjem komande **lexerGen** iz Ant skripte. Ova komanda će takođe obrisati postojeće automatski izgenerisane Java klasa.

**Generisanje klasa parsera** vrši se pokretanjem komande **repackage** iz Ant skripte. Pored toga, najpre će se ponovo izgenerisati i klase leksera komandom iznad. Da bi se generisane klase takođe i prevele, može se alternativno koristiti komanda **compile**.

**Prevođenje koda kompajlerom** vrši se pokretanjem Java klase **rs.ac.bg.etf.pp1.Compiler**. Ova klasa zahteva 2 parametara: prvi označava putanju do fajla sa izvornim kodom, dok drugi označava putanju gde će se izgenerisani bajtkod sačuvati.

**Pokretanje dobijenog koda** vrši se pokretanjem Java klase **rs.etf.pp1.mj.runtime.Run**. Ova klasa prima jedan parameter, a to je putanja to fajla sa bajtkodom. Može se dodati još jedan parameter, “-debug”, kojim se izvršavanje pokreće u debug režimu, čime se prikazuje trenutno stanje Sistema posle svake instrukcije.

Alternativno, može se koristiti komanda **runObj** iz Ant skripte.

# Opis test primera

Pored postojećih javnih test primera, priloženi su i sledeći test primeri:

* **syn\_errors.mj** – demonstrira oporavak od svih mogućih grešaka pri sintaksnoj analizi.
* **sem\_actpars.mj** – demonstrira hvatanje grešaka vezanih za broj i tip stvarnih parametara pri pozivu funkcije.
* **sem\_basictypes.mj** – demonstrira hvatanje grešaka prilikom neispravnih tipova podataka u aritmetičkim operacijama, čitanju i pisanju, kao i pri kreiranju nizova i hvatanju njihovih elemenata.
* **sem\_cond.mj** – demonstrira hvatanje grešaka prilikom neispravnosti uslovnih izraza.
* **sem\_const.mj** – demonstrira hvatanje grešaka prilikom deklarisanja konstanti.
* **sem\_constructor.mj** – demonstrira hvatanje grešaka prilikom deklarisanja konstruktora klase.
* **sem\_duplicate.mj** – demonstrira hvatanje grešaka prilikom korišćenja istog imena više puta u istom opsegu.
* **sem\_extends.mj** – demonstrira hvatanje grešaka prilikom neispravnog izvođenja iz klase.
* **sem\_loop.mj –** demonstrira hvatanje grešaka prilikom korišćenja break i continue naredbi izvan petlji.
* **sem\_nomain.mj** – demonstrira hvatanje grešaka prilikom nepostojanja main funkcije.
* **sem\_nonexistent.mj** – demonstrira hvatanje grešaka prilikom korišćenja nepostojećih imena.
* **sem\_return.mj** – demonstrira hvatanje grešaka prilikom neispravnih povratnih vrednosti u metodama.
* **sem\_super.mj** – demonstrira hvatanje grešaka prilikom neispravnih super poziva.
* **sem\_wrongtype.mj** – demonstrira hvatanje grešaka prilikom neispravnih tipova pri dodeli.

Uz sve primere dolaze i odgovarajući .out i .err fajlovi, sa ispisom sa standardnog izlaza i izlaza za greške.

# Opis klasa

Automatski generisane klase neće biti prikazane.

* **ClassTree** – predstavlja stablo klasa. Čvor stabla sadrži ime, pokazivač na objekat tipa klase u tabeli simbola, kao i listu pokazivača na svoju decu. tj klase izvedene iz nje.
* **CodeGenerator** – klasa u kojoj se obavlja generisanje koda. Posećuje čvorove zadatog apstraktnog sintaksnog stabla (koje je pre toga prošlo kroz semantičku analizu) i generiše kod na osnovu njega.
* **Compiler** – glavna klasa kompajlera. Kontroliše tok svih faza prevođenja. Upustvo za korišćenje je dato ranije u tekstu.
* **ExtendenDumpSymbolTableVisitor** – proširena verzija klase **DumpSymbolTableVisitor** iz tabele simbola, prilagođena potrebama i zahtevima projektnog zadatka.
* **ExtendedTab** – klasa sa statičkim podacima i funkcijama koje proširuju inicijalizaciju tabele simbola.
* **CounterVisitor** – klasa posetilac koja broji količinu određene vrste čvorova u apstraktnom sintaksnom stablu. Postoje dve potklase: **FormParamCounter** koja broji formalne parameter u datoj metodi, i **VarCounter** koja broji lokalne promenljive u datoj metodi.
* **SemanticAnalyzer** – klasa u kojoj se obavlja semantička analiza. Posećuje čvorove zadatog apstraktnog sintaksnog stabla, dopunjava je podacima neophodnim za generisanje koda, i proverava sve neophodne uslove.