Elektrotehnički fakultet u Beogradu Katedra za računarsku tehniku i informatiku

*Predmet:* Programski prevodioci 1 *Nastavnik:* dr Dragan Bojić, redovni profesor *Asistenti:* mast. inž. Maja Vukasović,

mast. inž. Mihajlo Ogrizović

*Školska:* 2022/2023.

*Student:* Anđela Glišović 2019/521

Projekat

**– Kompajler za Mikrojavu –**

# Kratak 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.

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

# Komande za generisanje java koda alatima

U korenu projekta nalazi se build fajl (*build.xml*) koji je prilagođen tako da postoje ciljevi za sve potrebne komande.

Za leksičku analizu koristi se specifikacija leksičkog analizatora zadata u fajlu “*spec/mjlexer.flex*“. Njen cilj je u build fajlu nazvan *lexerGen* i njegovim pokretanjem se generiše komanda   
*java -jar lib/Jflex -d ./src/rs/ac/bg/etf/pp1 spec/mjlexer.flex.*

Za sintaksnu analizu koristi se alat AST-CUP. Alat konstruiše LARL(1) parser na osnovu cup fajla “*spec/mjparser.cup*”. Cilj za pokretanje generisanja parsera je u build fajlu nazvan *parserGen* i njegovim pokretanjem se generiše komanda  
*java -jar lib/cup\_v10k.jar -destdir src/rs/ac/bg/etf/pp1 -ast src.rs.ac.bg.etf.pp1.ast -parser MJParser -buildtree spec/mjparser.cup* .  
Generiše se klasa MJParser.java i kompletno sintaksno stablo.

Kompajliranje svih izgenerisanih i uvedenih klasa može se izvršiti komandom koju poziva *compile* cilj iz build fajla.

# Prevođenje koda kompajlerom. Pokretanje i testiranje rešenja

Prevođenje mikrojava programa vrši se pokretanjem klase *Compiler* komandom čiji je prvi argument putanja do datog mikrojava program koji je potrebno prevesti, a drugi argument putanja do izlaznog *obj* fajla koji se može izvršavati na MikroJava Virtuelnoj Mašini. Klasa *Compiler* prvo vrši semantički analizu prolaskom kroz izgenerisano sintaksno stablo koristeći klasu *SemanticAnalyzator*, a zatim ako je analiza bila uspešna vrši generisanje koda prolazeći stablo koristeći klasu *CodeGenerator.*

Dobijeni izlazni objektni fajl može se pokrenuti na MJVM koristeći biblioteke *diasm* i *Run*.

# Opis priloženih test primera

Primer “*test/test02.mj*” je program koji sadrži sve gramatički podržane konstukcije, kao i instrukcije koje nisu podržane, a koje treba otkriti semantičkom analizom. Kompajliranjem datog primera klasom *Compiler* možemo videti primere urađenog oporavka od grešaka za globalne promenljive, kao i za operaciju dodeljivanja vrednosti. Takođe, možemo videti ispisane sve vrste semantičkih grešaka.

Testovi “*test/test301.mj*” i “*test/test01.mj*” su primerni sintaksno i semantički ispravnih programa čiji *obj* možemo pokrenuti.

# Opis novouvedenih klasa

Klasa *Compiler* instancira klase *SemanticAnalyzator* i *CodeGenerator* u cilju generisanja objektnog fajla. *SemanticAnalyzator* proširuje klasu *VisitorAdaptor* i implementira metode za obilazak sintaksnog stabla. Nakon završene semantičke analize generisana je i tabela simbola programa. *CodeGenerator* proširuje klasu *VisitorAdaptor* i implementira metode za obilazak sintaksnog stabla. Nakon završene analize se u klasi *Code* nalazi generisani kod, koji kompajler upisuje u objektni fajl pozivanjem funkcije *Code.write*().