SIMPLE

Team 9

Subhradeep Biswas (sbiswa24)

Prashansa(pprasha2)

Muhammad Sami(msami2)

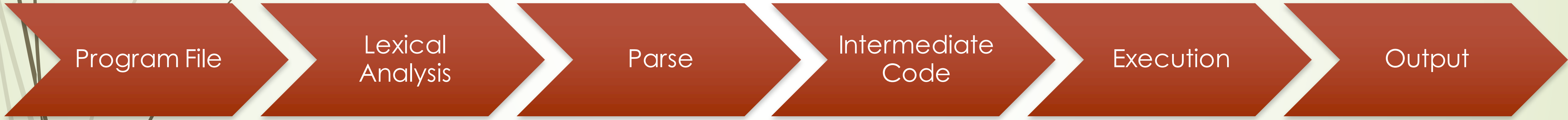
Basic features of language….

- Integer Type
- Boolean Type
- Looping
- All Arithmetic operations (+, -, *, /,%)
- Precedence

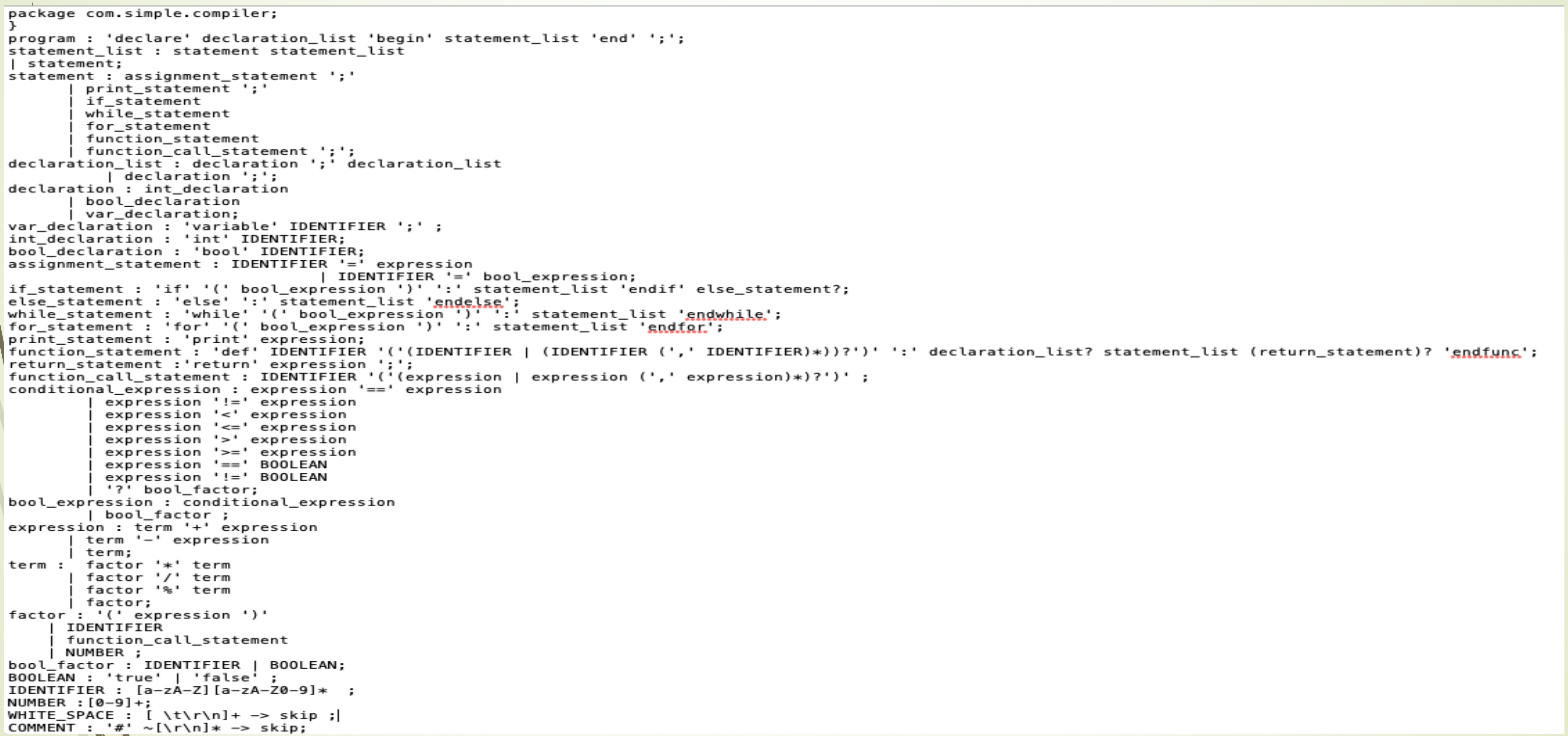
Extra Features of Language(Extra Credits)

- Nested Loops
- Functions
- Recursion
- Conditional expressions
- Boolean expression

Language Control Flow

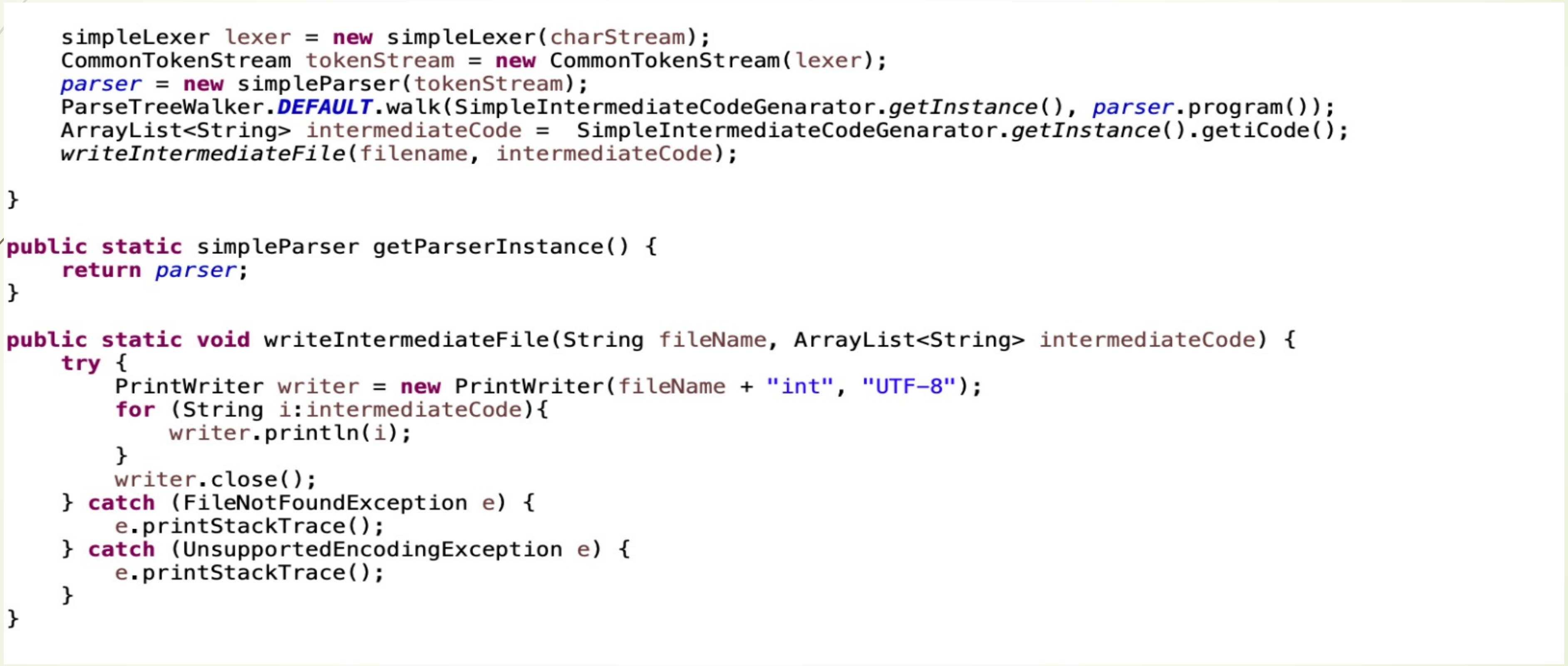


Grammer



```
package com.simple.compiler;
}
program : 'declare' declaration_list 'begin' statement_list 'end' ';';
statement_list : statement statement_list
| statement;
statement : assignment_statement ';'
        | print_statement ';'
        | if_statement
        | while_statement
        | for_statement
        | function_statement
        | function_call_statement ';';
declaration_list : declaration ';' declaration_list
            | declaration ';';
declaration : int_declaration
        | bool_declaration
        | var_declaration;
var_declaration : 'variable' IDENTIFIER ';' ;
int_declaration : 'int' IDENTIFIER;
bool_declaration : 'bool' IDENTIFIER;
assignment_statement : IDENTIFIER '=' expression
                                | IDENTIFIER '=' bool_expression;
if_statement : 'if' '(' bool_expression ')' ':' statement_list 'endif' else_statement?;
else_statement : 'else' ':' statement_list 'endelse';
while_statement : 'while' '(' bool_expression ')' ':' statement_list 'endwhile';
for_statement : 'for' '(' bool_expression ')' ':' statement_list 'endfor';
print_statement : 'print' expression;
function_statement : 'def' IDENTIFIER '('(IDENTIFIER | (IDENTIFIER (',' IDENTIFIER)*))?')' ':' declaration_list? statement_list (return_statement)? 'endfunc';
return_statement :'return' expression ';';
function_call_statement : IDENTIFIER '('(expression | expression (',' expression)*)?')' ;
conditional_expression : expression '==' expression
          | expression '!=' expression
          | expression '<' expression
          | expression '<=' expression
          | expression '>' expression
          | expression '>=' expression
          | expression '==' BOOLEAN
          | expression '!=' BOOLEAN
          | '?' bool_factor;
bool_expression : conditional_expression
          | bool_factor ;
expression : term '+' expression
        | term '-' expression
        | term;
term :  factor '*' term
        | factor '/' term
        | factor '%' term
        | factor;
factor : '(' expression ')'
      | IDENTIFIER
      | function_call_statement
      | NUMBER ;
bool_factor : IDENTIFIER | BOOLEAN;
BOOLEAN : 'true' | 'false' ;
IDENTIFIER : [a-zA-Z][a-zA-Z0-9]*   ;
NUMBER :[0-9]+;
WHITE_SPACE : [ \t\r\n]+ -> skip ;|
COMMENT : '#' ~[\r\n]* -> skip;
```

Compiler



```
    simpleLexer lexer = new simpleLexer(charStream);
    CommonTokenStream tokenStream = new CommonTokenStream(lexer);
    parser = new simpleParser(tokenStream);
    ParseTreeWalker.DEFAULT.walk(SimpleIntermediateCodeGenarator.getInstance(), parser.program());
    ArrayList<String> intermediateCode =  SimpleIntermediateCodeGenarator.getInstance().getiCode();
    writeIntermediateFile(filename, intermediateCode);

}

public static simpleParser getParserInstance() {
    return parser;
}

public static void writeIntermediateFile(String fileName, ArrayList<String> intermediateCode) {
    try {
        PrintWriter writer = new PrintWriter(fileName + "int", "UTF-8");
        for (String i:intermediateCode){
            writer.println(i);
        }
        writer.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
```

Syntactical Structure

```
DECL

Int a;
Int b;

Begin
a = 1;
b = 2;
Print a;
Print b;

End;
```

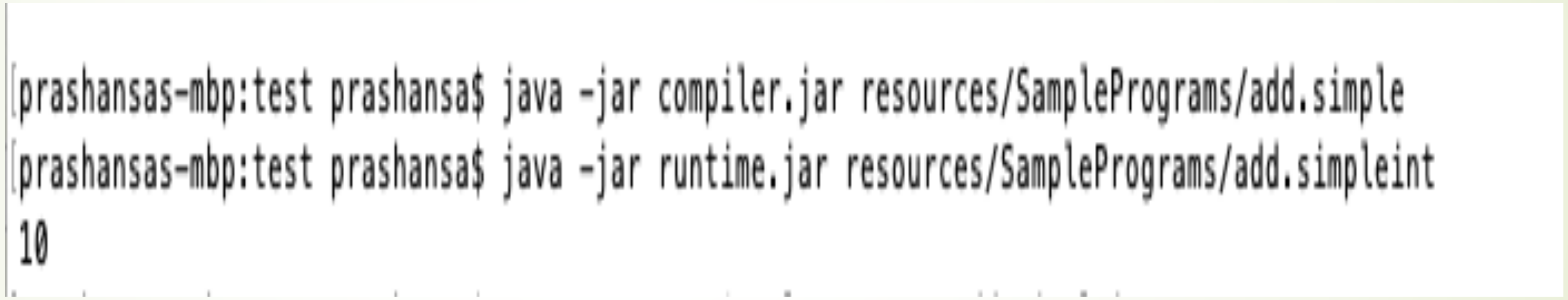
Sample Programs

- Addition
- Functions
- Precedence
- Recursion

Code and Intermediate Code Addition

CODE	Intermediate Code
declare	INT a
	INT b
int a;	INT c
int b;	PUSH 8
int c;	ASSIGNMENT a
	PUSH 2
	ASSIGNMENT b
begin	LOAD a
	LOAD b
a = 8;	ADDITION
b = 2;	ASSIGNMENT c
c = a + b;	LOAD c
print c;	PRINT

Compiling and Execution Addition



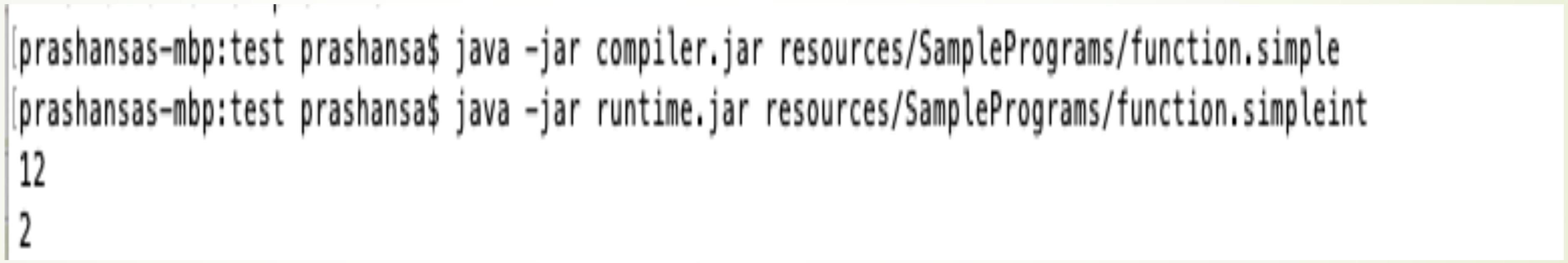
```
prashansas-mbp:test prashansa$ java -jar compiler.jar resources/SamplePrograms/add.simple
prashansas-mbp:test prashansa$ java -jar runtime.jar resources/SamplePrograms/add.simpleint
10
```

Code and Intermediate Code Function

Code	Intermediate Code	

int a;
int b;
int c;
int d;
int z;
begin
a = 1;
b = 2;
c = 3;
d = 4;
z = 0;
print MUL(c,d);
print MUL(a,b);
def MUL (x,y):
        z = x * y;
        return z;
endfunc
	INT a
	INT b
	INT c
	INT d
	INT z
	PUSH 1
	ASSIGNMENT a
	PUSH 2
	ASSIGNMENT b
	PUSH 3
	ASSIGNMENT c
	PUSH 4
	ASSIGNMENT d
	PUSH 0
	ASSIGNMENT z
	LOAD c
	LOAD d
		FUNCTION CALL_MUL
		PRINT
		LOAD a
		LOAD b
		FUNCTION CALL_MUL
		PRINT
		FUNCTION DECLARE_MUL
		FUNCTION_PARAM #MULx
		#MULy
		LOAD #MULx
		LOAD #MULy
		MULTIPLY
		ASSIGNMENT #MULz
		LOAD #MULz
		RETURN
		FUNCTION END_MUL

Compiling and Execution Function

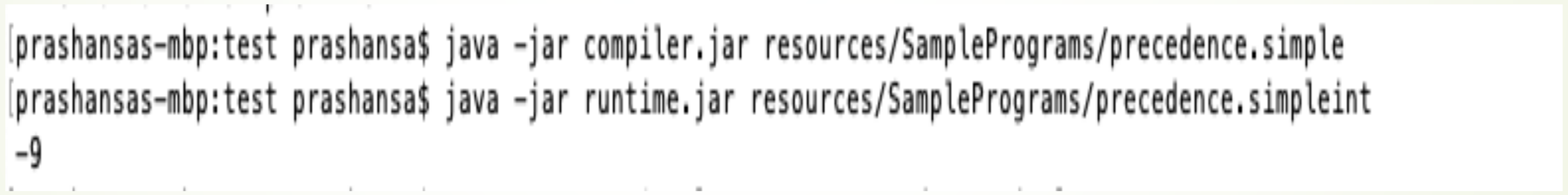


```
prashansas-mbp:test prashansa$ java -jar compiler.jar resources/SamplePrograms/function.simple
prashansas-mbp:test prashansa$ java -jar runtime.jar resources/SamplePrograms/function.simpleint
12
2
```

Code and Intermediate Code Precedence

Code	Intermediate Code	
int n;	INT n	ASSIGNMENT z
int b;	INT b	LOAD a
int c;	INT c	LOAD b
int d;	INT d	LOAD c
int z;	INT z	LOAD d
begin	PUSH 1	MULTIPLY
a = 1;	ASSIGNMENT a	SUBTRACTION
b = 2;	PUSH 2	ADDITION
c = 3;	ASSIGNMENT b	ASSIGNMENT z
d = 4;	PUSH 3	LOAD z
z = 0;	ASSIGNMENT c	PRINT
	PUSH 4	
z = a + b - c * d;	ASSIGNMENT d	
print z;	PUSH 0	

Compiling and Execution Precedence



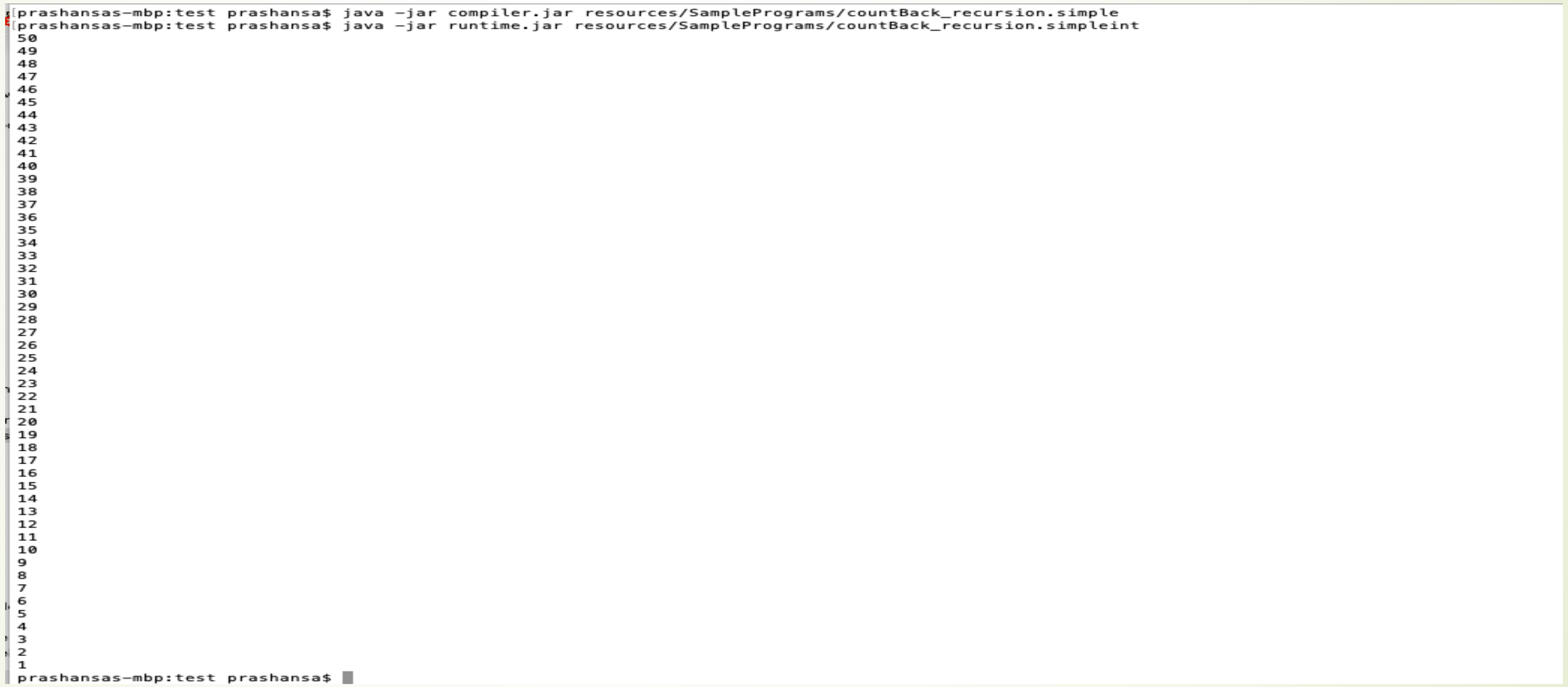
```
prashansas-mbp:test prashansa$ java -jar compiler.jar resources/SamplePrograms/precedence.simple
prashansas-mbp:test prashansa$ java -jar runtime.jar resources/SamplePrograms/precedence.simpleint
-9
```

Code and Intermediate Code
Recursion

Code	Intermediate Code	

declare
 	
int n;
begin
n = 50;
countBackwards(n);
def countBackwards(n):
if (n>0):
    print n;
    n = n-1;
    countBackwards(n);
endif
endfunc
	INT n
	PUSH 50
	ASSIGNMENT n
	LOAD n
	FUNCTION
	CALL_countBackwards
	FUNCTION
	DECLARE_countBackwards
	FUNCTION_PARAM
	#countBackwardsn
	IF_1
	LOAD #countBackwardsn
	PUSH 0
	GREATER_THAN
	CONDITION_END
		LOAD #countBackwardsn
		PRINT
		LOAD #countBackwardsn
		PUSH 1
		SUBTRACTION
		ASSIGNMENT
		#countBackwardsn
		LOAD #countBackwardsn
		FUNCTION
		CALL_countBackwards
		END IF_1
		FUNCTION
		END_countBackwards

Compiling and Execution Recursion



```
prashansas-mbp:test prashansa$ java -jar compiler.jar resources/SamplePrograms/countBack_recursion.simple
prashansas-mbp:test prashansa$ java -jar runtime.jar resources/SamplePrograms/countBack_recursion.simpleint
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
prashansas-mbp:test prashansa$
```

Future Work

- Other Data Types such as Float, Double, Char, String etc can be implemented.
- Collective data types such as Array, Linked list, Hash maps, trees and Graphs can be implemented to try complex algorithms.
- Object Oriented Programming concepts could be implemented to incorporate concepts like inheritance, Polymorphism etc

Thank you!!