# CS 352 – Spring 2011
# Project 3a

Handed out: **Mar 1st, 2011**

Due: **Mar 27th, 2011 @ 11:59pm**

**Abstract:**

In this project, you will create a parser that constructs an abstract syntax tree (AST) while it parses user input. In order to do this, you will modify the given skeleton code to correctly represent the given grammar, which is a simplified version of the one in project two. Then you will add AST building instructions within that parser.

**Setting up required tools:**

Your setup for this project will be similar to that in project 2.

**Setting up your code skeleton:**

The code skeleton consists of **MiniJavaParser.jj** and support directories **syntaxtree** and **visitor.** All are compressed as **Project3a.zip** file.
(You are required to use this skeleton code as a starting point.)

**What you need to do:**

In MiniJavaParser.jj you will add functionality ***to parse tokens*** according to grammar specifications. JavaCC will produce the corresponding java code from the .jj file.
Main function will then takes an input file, tokenize it, and parse it based on your parser specification and print out corresponding AST.
All your work should be done in the MiniJavaParser.jj file. (You should not modify any other files)

**The output of your code:**

You are not required to output anything. If your program parses an input file cleanly and constructs the AST correctly, a call in main function will correctly print out your AST.

**Running your code skeleton (how TA will run the code);**

TA creates a fresh copy of the code skeleton
TA copies your MiniJavaParser.jj file over the same file in the code skeleton directory
TA goes into the directory containing MiniJavaParser.jj (the code skeleton directory)
TA runs:

```
javacc MiniJavaParser.jj
javac Parser/*.java
java Parser/MiniJavaParser testInputFile.txt
```

**Turnin:**

You should turn in the **MiniJavaParser.jj** file using the ***`turnin`*** command available on CS unix machines.
**To submit: turnin –c cs352 –p p3a MiniJavaParser.jj**
**To verify: turnin –v –c cs352**
(There will be a penalty for turning in incorrectly)

Any doubts concerning the project requirements should be raised by sending email to lib@purdue.edu. The TA may then discuss things with the instructor if necessary.

# BNF for MiniJava

## NON-TERMINALS

| | | |
|---|---|---|
| Goal | ::= | MainClass ( ClassDeclaration )* <EOF> |
| MainClass | ::= | **class** Identifier "{" **public static void main** "(" **String** "[" "]" Identifier ")" "{" Statement "}" "}" |
| ClassDeclaration | ::= | **class** Identifier ( **extends** Identifier )? "{" ( VarDeclaration \| MethodDeclaration )* "}" |
| VarDeclaration | ::= | Type Identifier ("," Identifier)*  ";" |
| MethodDeclaration | ::= | **public** Type Identifier "(" ( Type Identifier ( "," Type Identifier )* )? ")" "{" ( VarDeclaration \| Statement )* **return** Expression ";" "}" |
| Type | ::= | NameType |
| | \| | Type "[" "]" |
| NameType | ::= | **int** |
| | \| | **boolean** |
| | \| | Identifier |
| Statement | ::= | "{" ( Statement )* "}" |
| | \| | **if** "(" Expression ")" Statement ( **else** Statement )? |
| | \| | **while** "(" Expression ")" Statement |
| | \| | **System.out.println** "(" Expression ")" ";" |
| | \| | Identifier "=" Expression ";" |
| | \| | Identifier "[" Expression "]" "=" Expression ";" |
| Expression | ::= | Expression ("&&" \| "\|\|" \| "<" \| ">" \| "+" \| "-" \| "*" \| "/") Expression |
| | \| | Expression "[" Expression "]" |
| | \| | Expression "." "length" |
| | \| | Expression "." Identifier "(" ( Expression ( "," Expression )* )? ")" |
| | \| | <INTEGER_LITERAL> |
| | \| | **true** |
| | \| | **false** |
| | \| | Identifier |

```
                    |  this
                    |  new NameType "[" Expression "]" ( "[" "]" )*
                    |  new Identifier "(" ")"
                    |  "!" Expression
                    |  "(" Expression ")"
Identifier ::= <IDENTIFIER>
```