

## Type Checker

For this milestone you will extend your program from the previous milestone to type check MiniJava programs.

Your type checker will take the name of a text file on disk as an argument. The program will parse the file according to the syntax given in the Term Project Description. Your program will then type check the input file according to the usual Java type checking rules for expressions, assignments, statements, and declarations. In addition to these usual checks, your program will also check the additional MiniJava type checking rules given in the Term Project Description.

If the input file is ill typed, then your program should emit an error message indicating the sort of error that occurred. You do not need to indicate the position of the error in the file, though you may certainly choose to do so.

I encourage you to use the approach discussed in class of building an abstract syntax tree (AST) for the input file. Use a tree-walk on this AST to perform the typechecking. You may find it useful to use a symbolic debugger to traverse your generated ASTs. Another approach is to write code to “pretty print” your ASTs. I use JUNIT to test my type checking code, because regression errors are common in this sort of code.

### HINTS AND SAMPLE CODE

Many students will find this milestone to be significantly more challenging than the previous ones. You will not be able to complete the milestone if you leave it to the last minute.

To lessen the burden, I will be posting hints, sample code, and sample test cases to the course Angel site. Please start your work on this milestone early, take advantage of the posted resources, and ask for help as needed.

### EXAMPLES

Below are a few of examples of the sort of output that your program might emit:

#### Class Redeclaration

Input: `class Foo { public static void main(String[] args) { }} class Foo { }`

Output: Class named Foo already exists.

#### Method Overloading

Input: `class Foo { public static void main(String[] args) { }}  
class Bar { public Bar self() { return this; } }  
class Baz extends Bar { public int self() { return 0; } }`

Output: Cannot overload methods. Method self has different type signature than inherited method of the same name.

Actual return type of method self does not match declared type.

## Field Redeclaration

Input: 

```
class Foo { public static void main(String[] args) { }}
class Bar { int x; Bar x; }
```

Output: The class variable x is already declared. Redeclaration and shadowing are not allowed.

## DELIVERABLES

### Sample Input

You must submit 8 sample input files, **by class time at least three days before** the milestone deadline. *Two of the sample inputs should be valid MiniJava programs. The other six should include at least one type error.* I strongly encourage you to:

- submit small test cases focusing on a particular type error,
- run your parser against the test cases to make sure they are valid MiniJava programs, and
- run `javac` against the test cases to identify type errors.

Your sample input files should be wrapped in a zip or rar archive and uploaded to the drop box on Angel. Alternatively, you may commit the sample input files to your Subversion repository and let me know where to look for them. Please limit your sample inputs to no more than 250 lines and no more than 80 characters per line, though as indicated above, smaller is better. Note that you do not have to submit sample output at this time.

I will select two sample inputs from each team and give the set of these to every team at least two days before the milestone deadline. I will give the remaining sample inputs to every team at class time on the milestone deadline. (We will generally not have class on milestone days. The sample inputs will be posted on Angel.)

### Sample Output and Code

By midnight on the milestone deadline you must upload to the dropbox on Angel a zip or rar archive containing:

- the output of your program for each sample input, one output file per input file, with the same name as the input file but the extension changed to “.out”
- the source code for your program

Alternatively, you may commit the sample output and source code to your Subversion repository and let me know where to look for it.

### Grading

Your grade will be based on whether you submit the required sample inputs and the percentage of the sample input files that your program processes correctly.

I'll award bonus points for sample inputs that expose bugs in other teams' programs.