*"Hey guys, we're making a new language!"*

*"k."*

# k-AWK (kay-awk)

## The Testing Language

Albert Cui, Karen Nan,
Michael Raimi, Mei-Vern Then

## Overview: Motivation

- Automated testing for quality assurance
- Test-driven development
- Design software in a robust manner

## Overview: k-AWK

- Checks for predefined statements within each struct (asserts)
- When called or initialized, all assertions evaluated to `true` allow program to continue
- *unit* features attached to functions check output in test mode

## Tutorial: Program Execution

- Extension for k-AWK programs: `.k`

- Run make to create `code_gen`:

```
$ ./code_gen foobar.k
```

- To compile and run, use the test script. Outputs to stdout and to a .txt file.

```
$ ./run.sh foobar.k
```

# Tutorial: Asserts

- Similar to `if` statements, can only be used in `structs`

- Starts with `@` symbol, followed by an expression and a block of statements:

```
@(k < 100) { print("k is >= 100!"); }
```

- Asserts are evaluated whenever a variable in the expression is changed

- If k is less than 100, the program continues. If not, the print statement within the attached block is executed.

# Tutorial: Units

```
unit:foo(hi):equals(1):accept;
```

- Four parts, separated by single colon:
  - unit: indicates the start of the unit test call
  - foo(hi): indicates the function to call and its arguments
  - equals(1): a logical expression that matches its argument to the return value of the function
  - accept: indicates whether or not a test should pass if a `true` value is returned from the logical expression (above)
    - reject keyword that tells a unit test to fail if the logical expression returns `true`

## Tutorial: Built-In Functions

- `print(10);`
  - Takes in one string or integer argument
  - Prints to stdout
- `exit(“foobar”);`
  - Takes in one string argument
  - Prints string to stdout, then exits program

## Example Programs: `hello_world.k`

```
void main() {
   print ("Hello, world! k-Awk says hi);
}
```

- must have main function of type `void`, takes no arguments
- uses built-in print function to print string to stdout

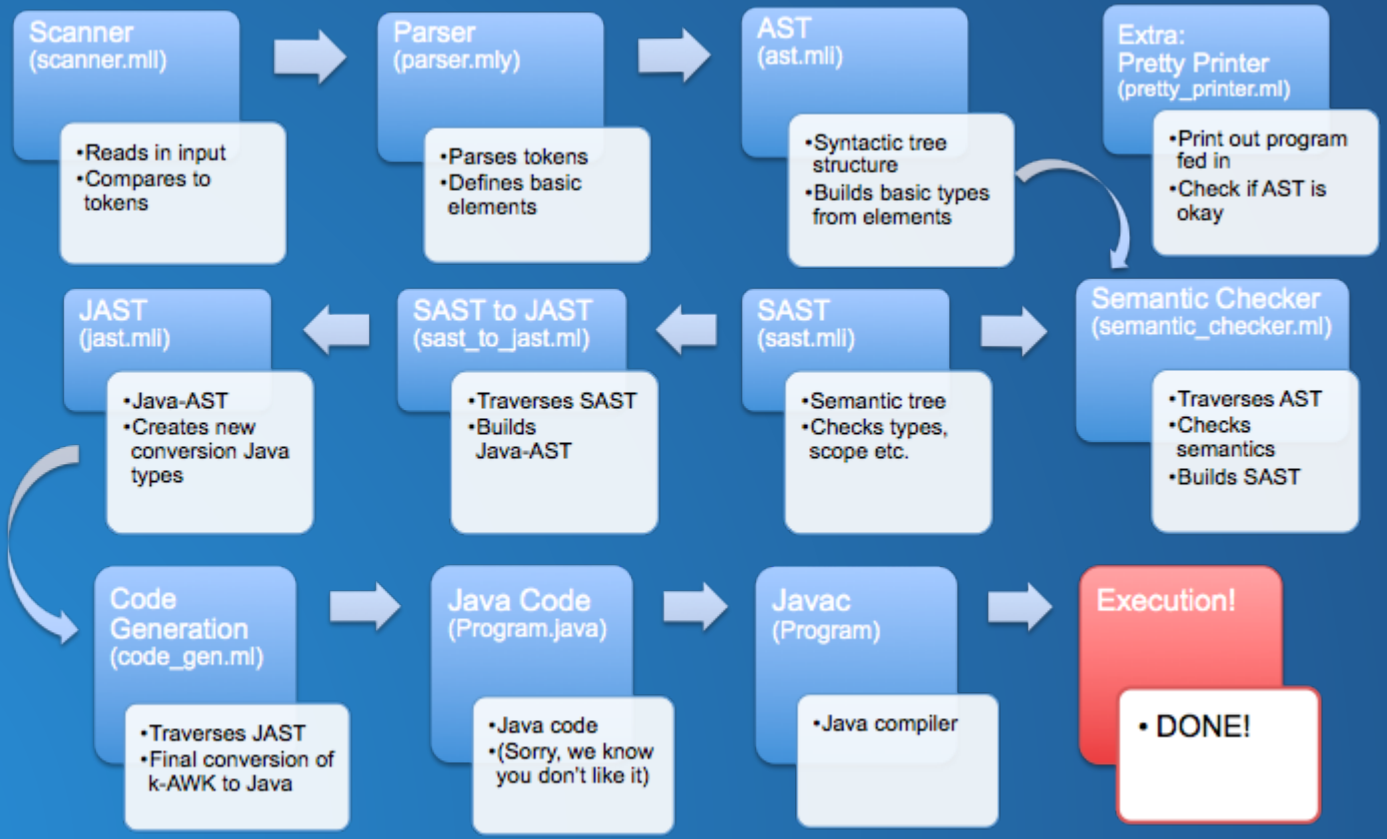
## Example Programs: `gcd.k`

- One function, called by main with unit test
- Functions must be defined before main to be used
- Unit tests call other functions
  - Prints whether the test passes or fails, with calls and values

## Example Programs: `99_bottles.k`

- Main calls function with int value
    - Function creates instance of struct
        - Runs struct and uses assert to decrement
- Prints out statements specified in asserts, prints outcome of unit test

# Language Implementation



## Lessons Learned

- Prioritize:
  - Too much time spent on the pretty printer
- Move decisively but consider future implications
- Better breakdown of project into smaller chunks
- Smaller, more incremental goals