DECAF

Hidy Han, JiaYan Hu, Kim Tao, Kylie Wu

Overview

DECAF - So Easy to Use You Won’t Even Need Caffeine

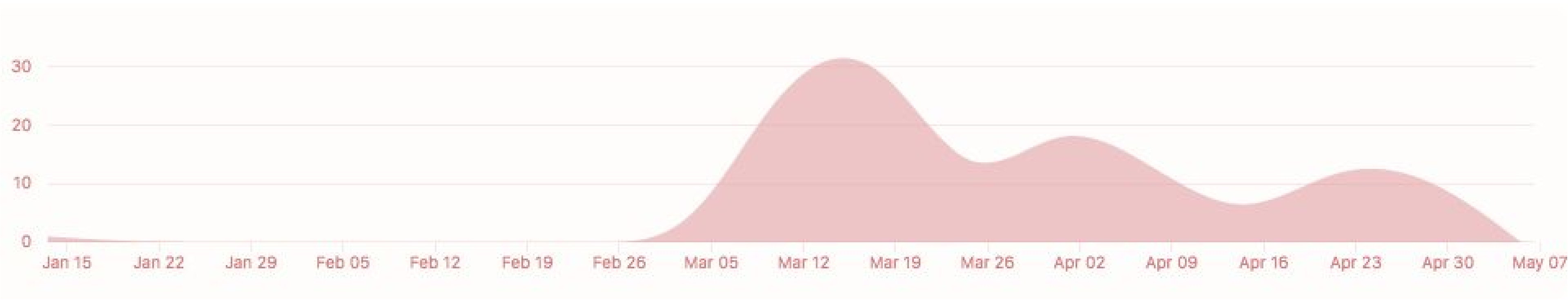
General-Purpose Programming Language, with core features extracted from Java and C.

- Top-level classes and functions
- Comprehensive C features
- Simple Object-Oriented functionality

Goals

- Safety:
  - DECAF is statically typed.
  - Explicit rather than implicit type casting.
- Familiarity:
  - Syntax designed to resemble that of Java and C
  - Great for both beginner and veteran programmers.

Project Timeline



Development Environment

LLVM

Bleeding Edge Compiler Technology

Syntax

Comments

```
// A single line comment

/*
    A multi-line comment
*/
```

Operators

```
+               // add
-               // subtract
*               // multiply
/               // divide
%               // modulus
and             // and
or              // or
not             // not
===             // eq.
!==             // neq.
<               // lt.
>               // gt.
<=              // leq.
>=              // geq.
<type>          // cast
```

Built-in Types

```
bool                // true, false
int                 // 4115
float               // 0.25
char                // ‘h’
string              // “Hello World”
array               // [int] arr = [int, 5]
```

Syntax

Arrays

```
[int] nums = [int, 5];

nums[0] = -5;
nums[1] = nums[0];

print_string(“nums[0] is now ”);
print_int(nums[0]);
```

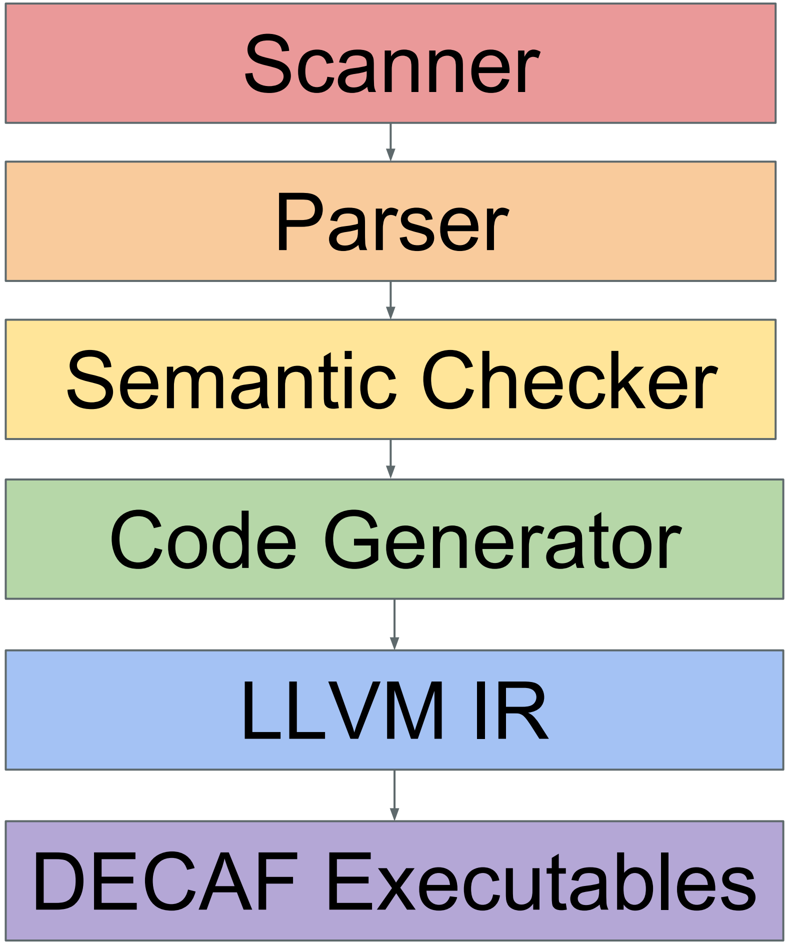
Control Flow

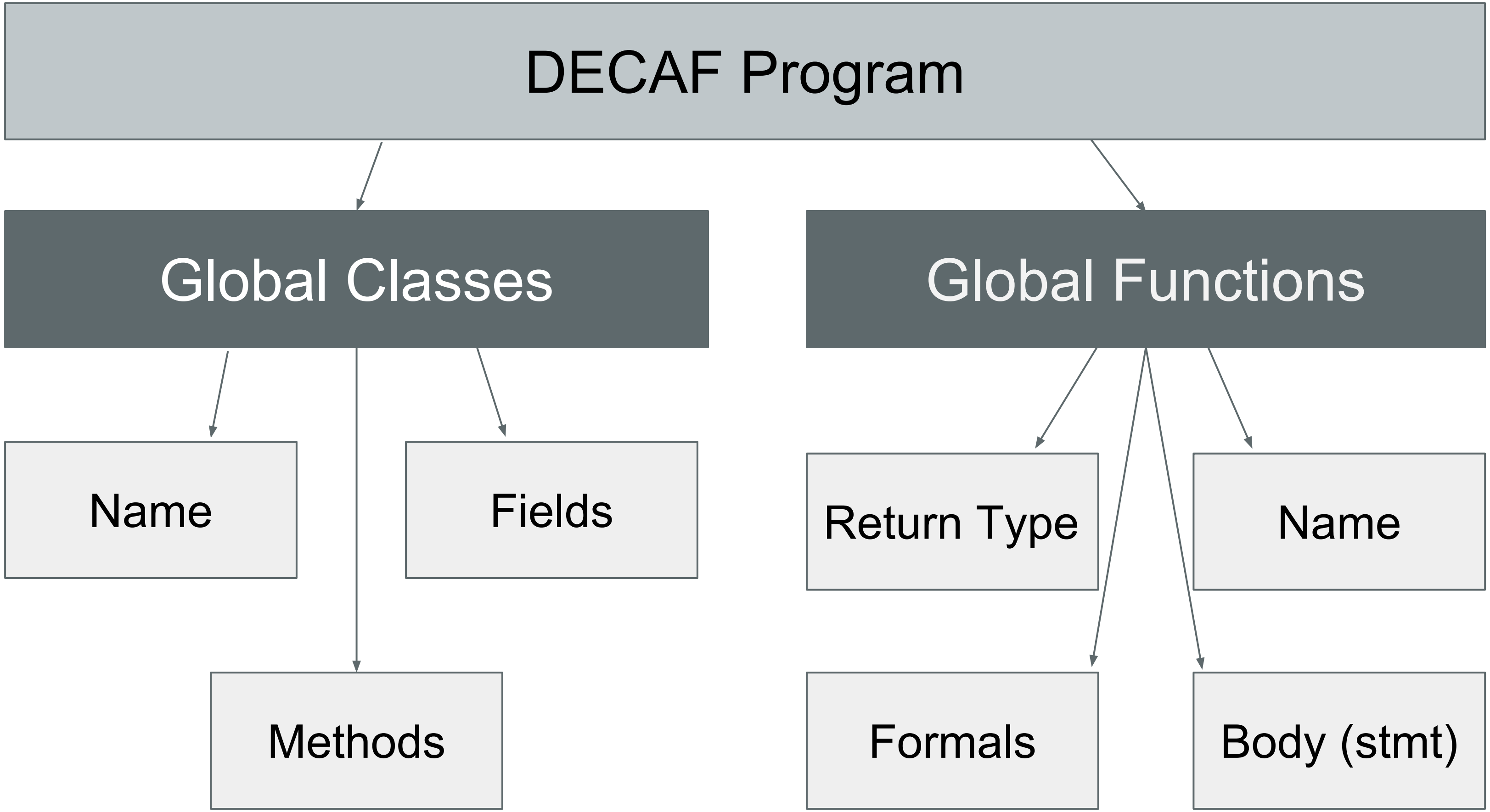
```
main() -> int {
        int x;
        while (x < 5) {
                x = x + 1;
                if (true) {
                        print_string("foo");
                        continue;
                }
                print_string("bar");
        }
        return 0;
}
```

Classes

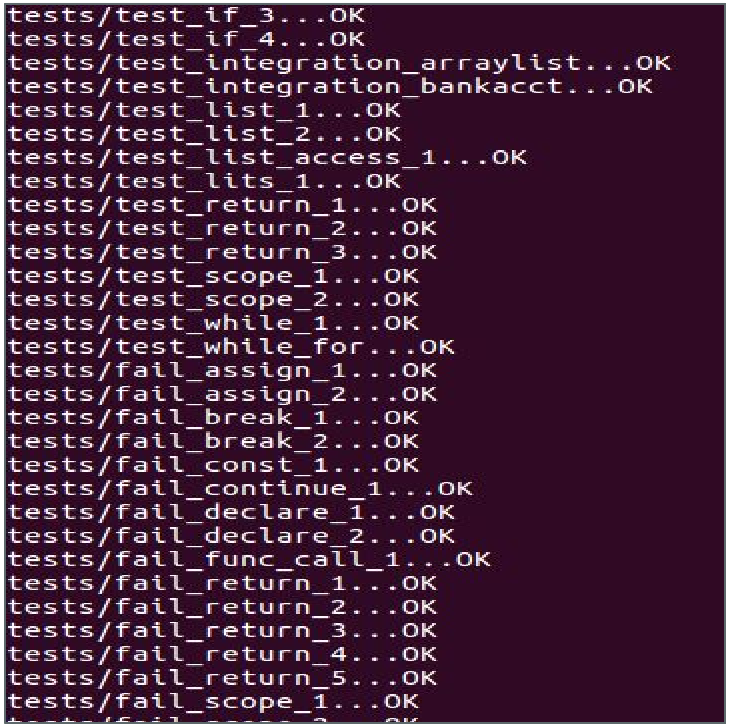
```
class Animal {
        string name;
        int legs;
        Animal(string n, int l) -> Animal {
                self.name = n;
                self.legs = l;
        }
        talk() -> void {
        print_string(self.name);
                print_string(" says hi\n");
        }
}
```

Architecture




DECAF Program
Global Classes
Global Functions
Name
Fields
Methods
Return Type
Name
Formals
Body (stmt)

Test Suite



```
tests/test_if_3...OK
tests/test_if_4...OK
tests/test_integration_arraylist...OK
tests/test_integration_bankacct...OK
tests/test_list_1...OK
tests/test_list_2...OK
tests/test_list_access_1...OK
tests/test_lits_1...OK
tests/test_return_1...OK
tests/test_return_2...OK
tests/test_return_3...OK
tests/test_scope_1...OK
tests/test_scope_2...OK
tests/test_while_1...OK
tests/test_while_for...OK
tests/fail_assign_1...OK
tests/fail_assign_2...OK
tests/fail_break_1...OK
tests/fail_break_2...OK
tests/fail_const_1...OK
tests/fail_continue_1...OK
tests/fail_declare_1...OK
tests/fail_declare_2...OK
tests/fail_func_call_1...OK
tests/fail_return_1...OK
tests/fail_return_2...OK
tests/fail_return_3...OK
tests/fail_return_4...OK
tests/fail_return_5...OK
tests/fail_scope_1...OK
```

- Automated in testall.sh
- Compares output with test_case.out
- New test cases added when new features are added

Demo