

LIVA

A Lite Version of Java



Shanqi Lu, Jiafei Song, Zihan Jiao, Yanan Zhang, Hyoyoon Kate Kim

Introduction

What we are looking for from LIVA

Simple

It is designed to let programmers, who are familiar with class-based languages, to feel comfortable with developing common algorithms like GCD. It is lite in the sense that it maintains some but not all features in Java.

Object-Oriented

It has a Java-like syntax and supports object-oriented paradigm and inheritance.

Portable

LIVA is a portable language and compiled down to LLVM.

Tools



GitHub

Version Control



VMware

Make Development Consistent



Ubuntu

Operating System

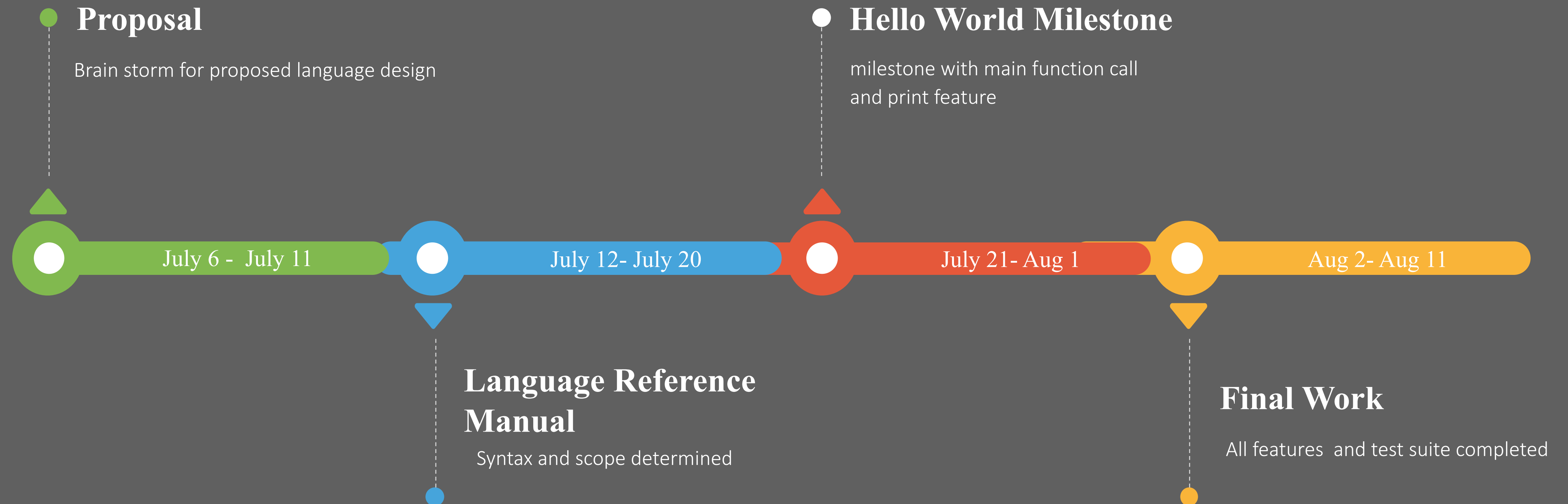


OS X

Operating System

Project Schedule

36 Days' Project !!!



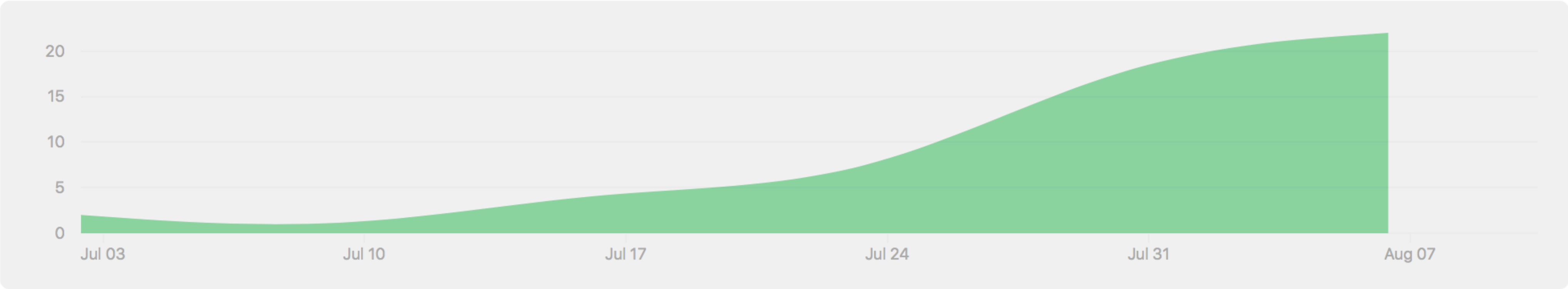
GitHub History



Jul 3, 2016 – Aug 11, 2016

Contributions: **Commits** ▾

Contributions to master, excluding merge commits



Features



LIVA

Types



Operators



Functions



Classes



Inheritance



Arrays



Loops



If-Else



Standard Library



Syntax



Type

```
int a = 1;
float b = 2.2;
char c = '3';
boolean d = true;

int[] x = new int[10];
float[] y = new float[10];
y[1] = 1.0;
```

Operators

'+'	{ PLUS }
'-'	{ MINUS }
'*'	{ TIMES }
'/'	{ DIVIDE }
'%'	{ MODULO }
'='	{ ASSIGN }
"=="	{ EQ }
"!="	{ NEQ }
'<'	{ LT }
"<="	{ LEQ }
">"	{ GT }
">="	{ GEQ }
"&"	{ AND }
" "	{ OR }
"!"	{ NOT }

Comments

```
/*This is a Liva comment!*/
```

Syntax



Control Flow

```
int i;
for (i=0; i<10; i=i+1){
    print(i);
}

while (i > 0){
    print(i);
    i = i - 1;
}

if (true){
    print(42);
} else {
    print(8);
}
```

Class

```
class myclass{
    int calc (int x, int y){
        int z;
        z = x + y;
        return (z);
    }
}

class subcls extends myclass{
    int b;
    constructor(int a){
        this.b = a;
    }
    int calc (int x, int y){
        int z;
        z = x - y;
        return (z);
    }
}
```

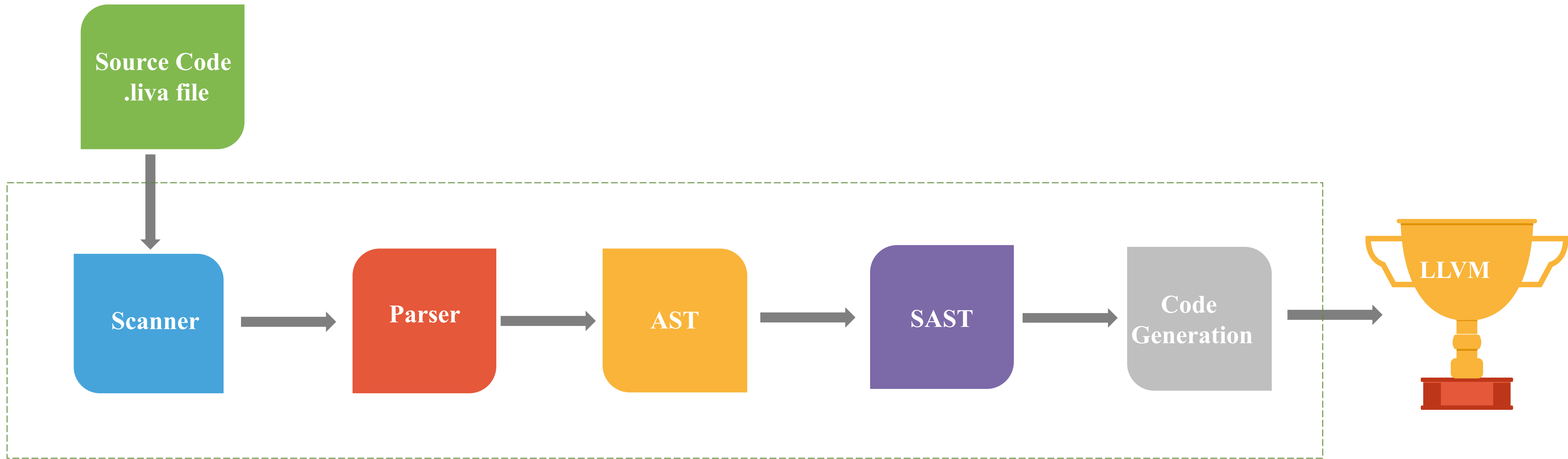
Object

```
class test {
    void main(){
        int x = 9;
        int y = 6;
        int z;

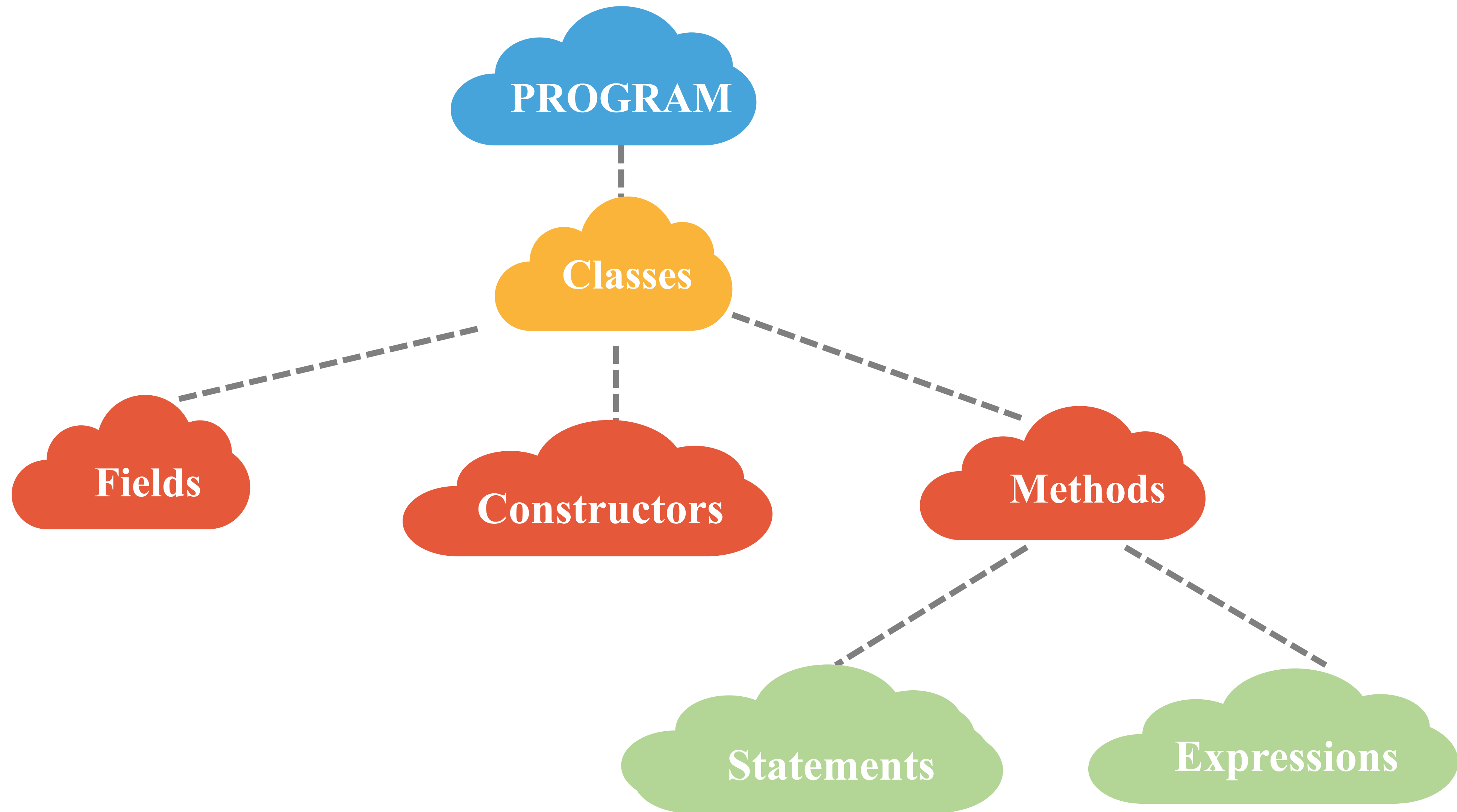
        class myclass obj =
new myclass();
        z = obj.calc(x, y);

        print ("z=",z);
    }
}
```

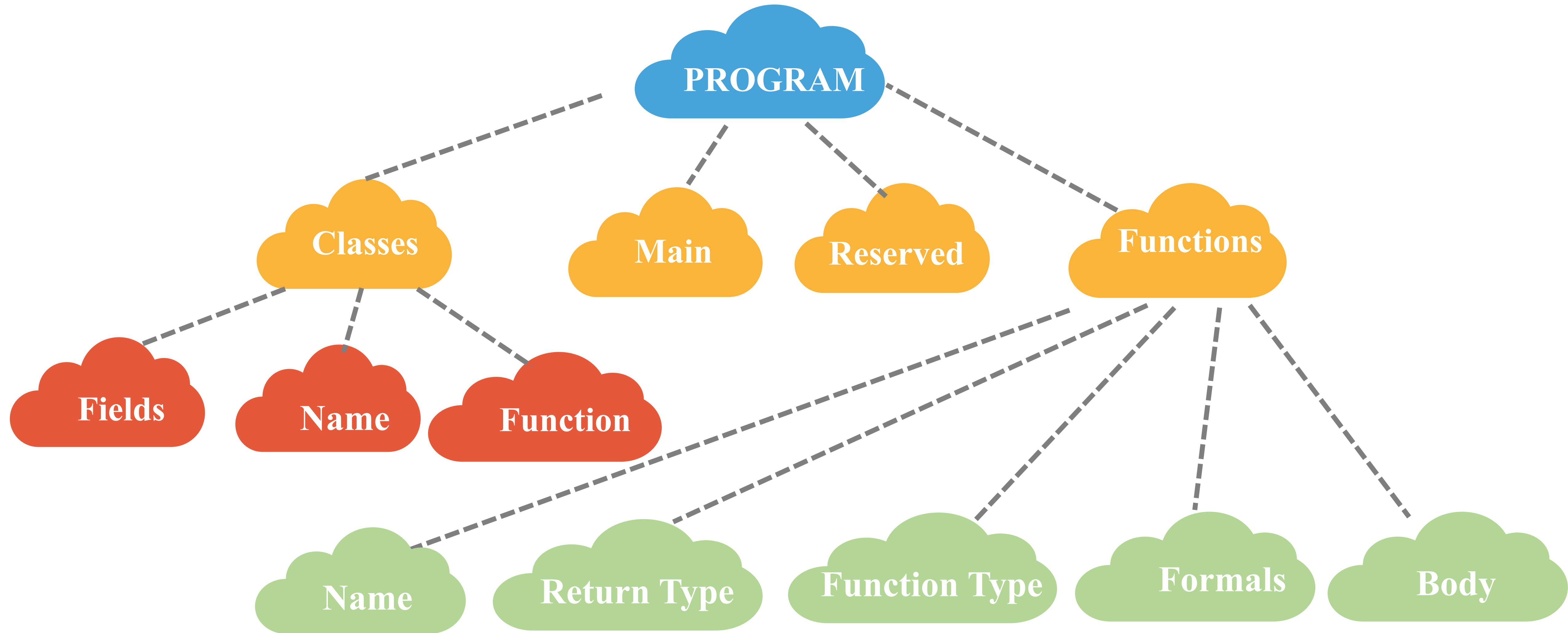

Architecture



AST



SAST

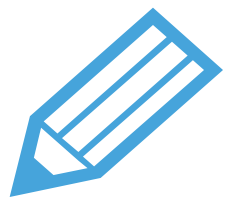


Testing



Test procedure

- Compile and run test-if1.liva
- Compare the output with test-if1.out
- If they are the same, done! Otherwise, find out the problems.



120 test files

- All passed!



Testall.sh

- Based on the test script of MicroC
- Test all files in a single command

Testing




```
jiafei@jiafei-virtual-machine: ~/OC
jiafei@jiafei-virtual-machine:~/OCa
jiafei@jiafei-virtual-machine:~/OCa
jiafei@jiafei-virtual-machine:~/OCa
test-add...OK
test-and...OK
test-arith...OK
test-array...OK
test-array_object...OK
test-comments...OK
test-constructor...OK
test-diff...OK
test-div...OK
test-equal...OK
test-fib...OK
test-for1...OK
test-for_nest...OK
test-function...OK
test-gcd...OK
test-geq...OK
test-gt...OK
test-hello...OK
test-hello2...OK
test-if1...OK
test-if_nest...OK
```


```
jiafei@jiafei-virtual-machine: ~/OCa
test-inheritance...OK
test-inheritance2...OK
test-leq...OK
test-lt...OK
test-mod...OK
test-mul...OK
test-nequal...OK
test-not...OK
test-obj...OK
test-or...OK
test-override...OK
test-sub...OK
test-while1...OK
test-while_for_nest...OK
test-while_nest...OK
fail-add...OK
fail-array_access...OK
fail-array_access2...OK
fail-array_init...OK
fail-diff...OK
fail-div...OK
fail-equal1...OK
fail-equal2...OK
fail-for1...OK
```

```
fail-function...OK
fail-function2...OK
fail-function3...OK
fail-hello...OK
fail-hello2...OK
fail-if1...OK
fail-mod...OK
fail-mul...OK
fail-not...OK
fail-obj_access...OK
fail-obj_access2...OK
fail-obj_access3...OK
fail-sub...OK
fail-while1...OK
jiafei@jiafei-virtual-machine:~/OCa
```


Testing





 test-add.liva


 test-add.out


 ...


 test-equal.liva


 test-equal.out


 test-function.liva


 test-function.out


 fail-add.err


 fail-add.liva


 fail-equal1.err


 fail-equal1.liva


 fail-equal2.err


 fail-equal2.liva


 fail-function.err

 fail-function.liva

 fail-function2.err

 fail-function2.liva

 fail-function3.err

 fail-function3.liva


Unit Test


Successful & Unsuccessful


Small pieces


Testing




 test-gcd.liva


 test-gcd.out


 test-geq.liva


 test-geq.out


—


 test-if_nest.liva

 test-if_nest.out

 test-while_for_nest.liva

 test-while_for_nest.out

 test-while_nest.liva

 test-while_nest.out

Integration Test

Small tests integrated into larger one

Lessons Learned



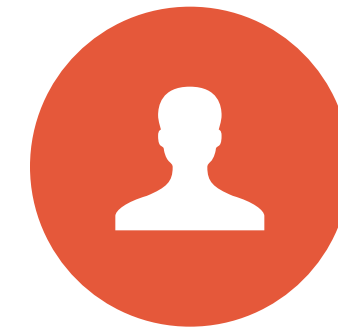
Time Management

Start the project early



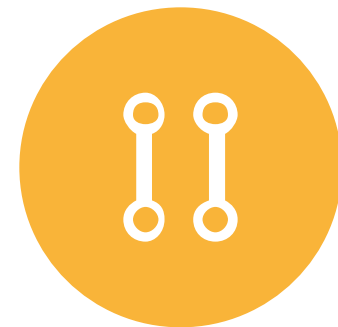
Cooperation

Teamwork and integration



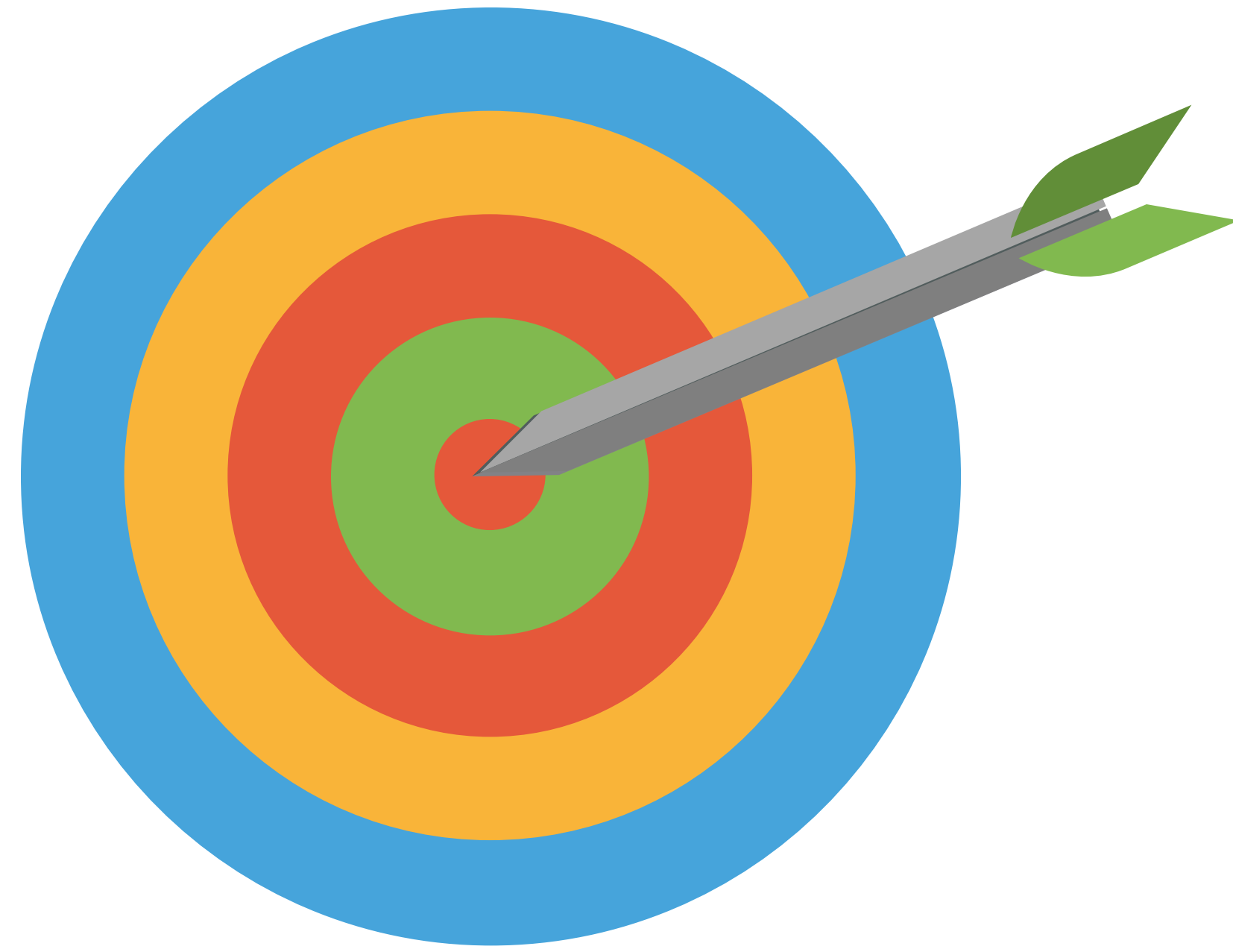
Communication

Avoid doing the same work



Software Tools

Efficiency improvement



DEMO