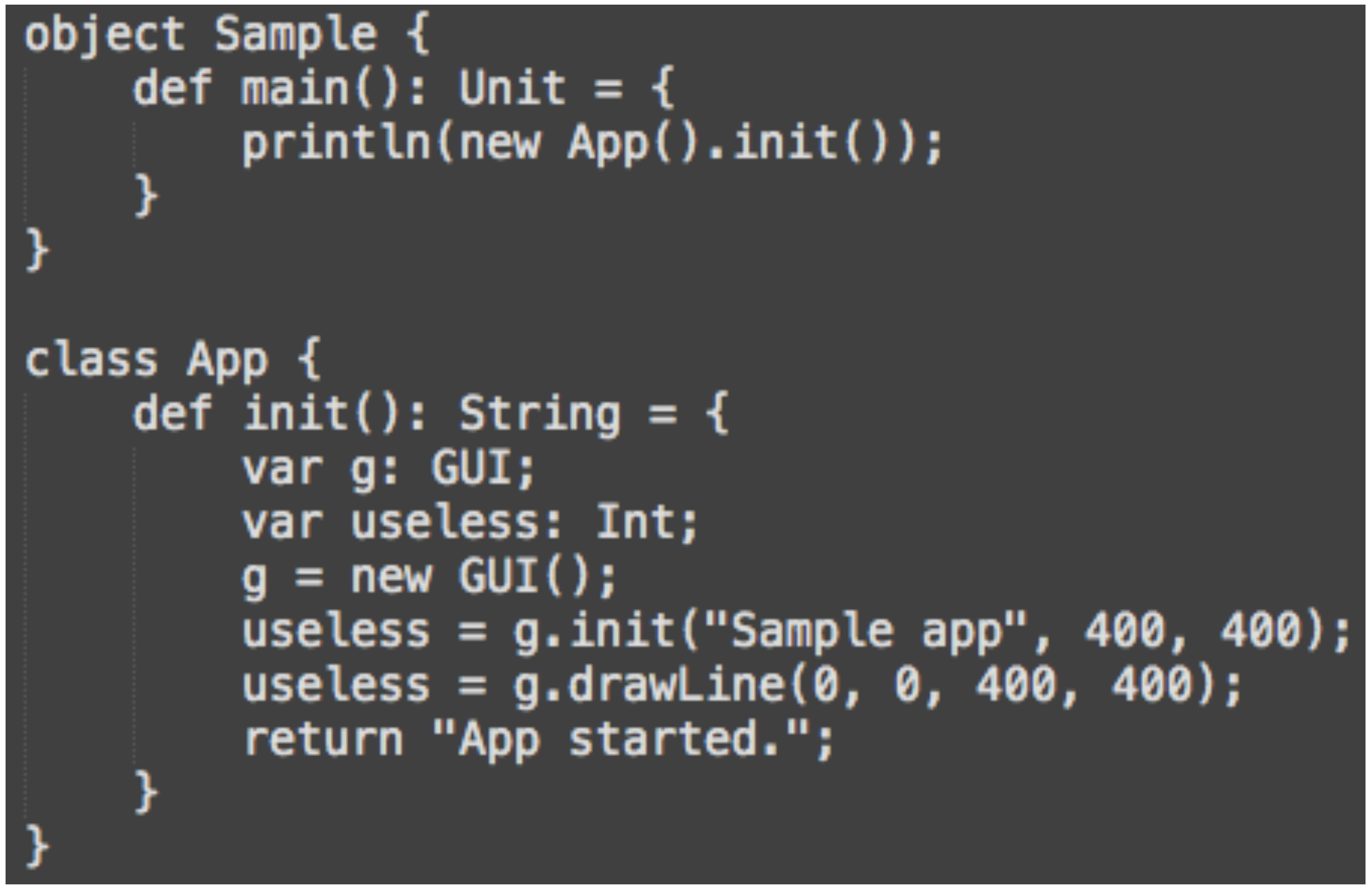
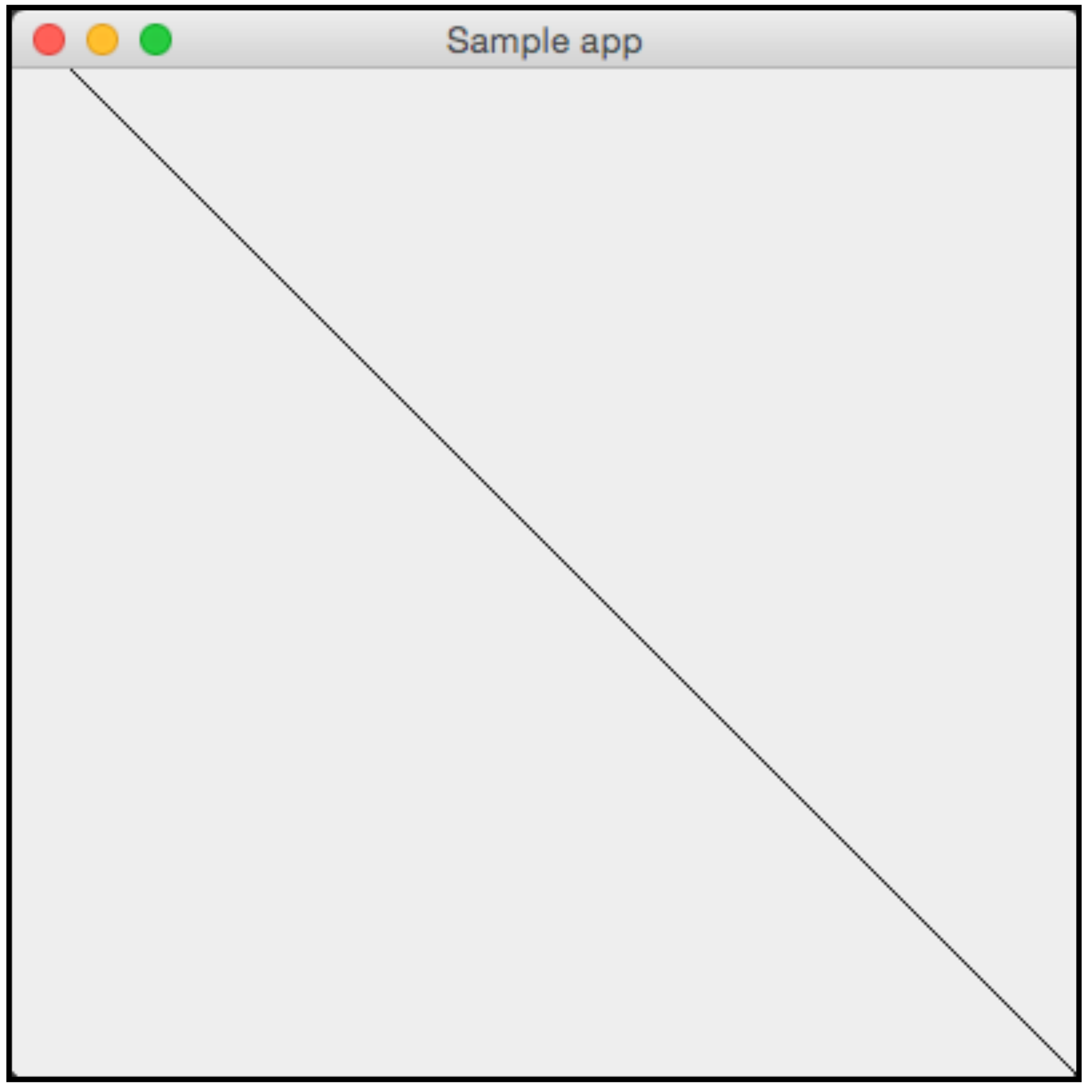
Sample program with ToolGUI



```
object Sample {
    def main(): Unit = {
        println(new App().init());
    }
}

class App {
    def init(): String = {
        var g: GUI;
        var useless: Int;
        g = new GUI();
        useless = g.init("Sample app", 400, 400);
        useless = g.drawLine(0, 0, 400, 400);
        return "App started.";
    }
}
```



GENERAL IDEA - What the programmer sees

A class GUI which, when instantiated and being called init() on it, creates:

- a frame to draw on
- a timer
- keyboard interface

Graphics!

User interaction!

Apps!

Games!

~~Money!~~

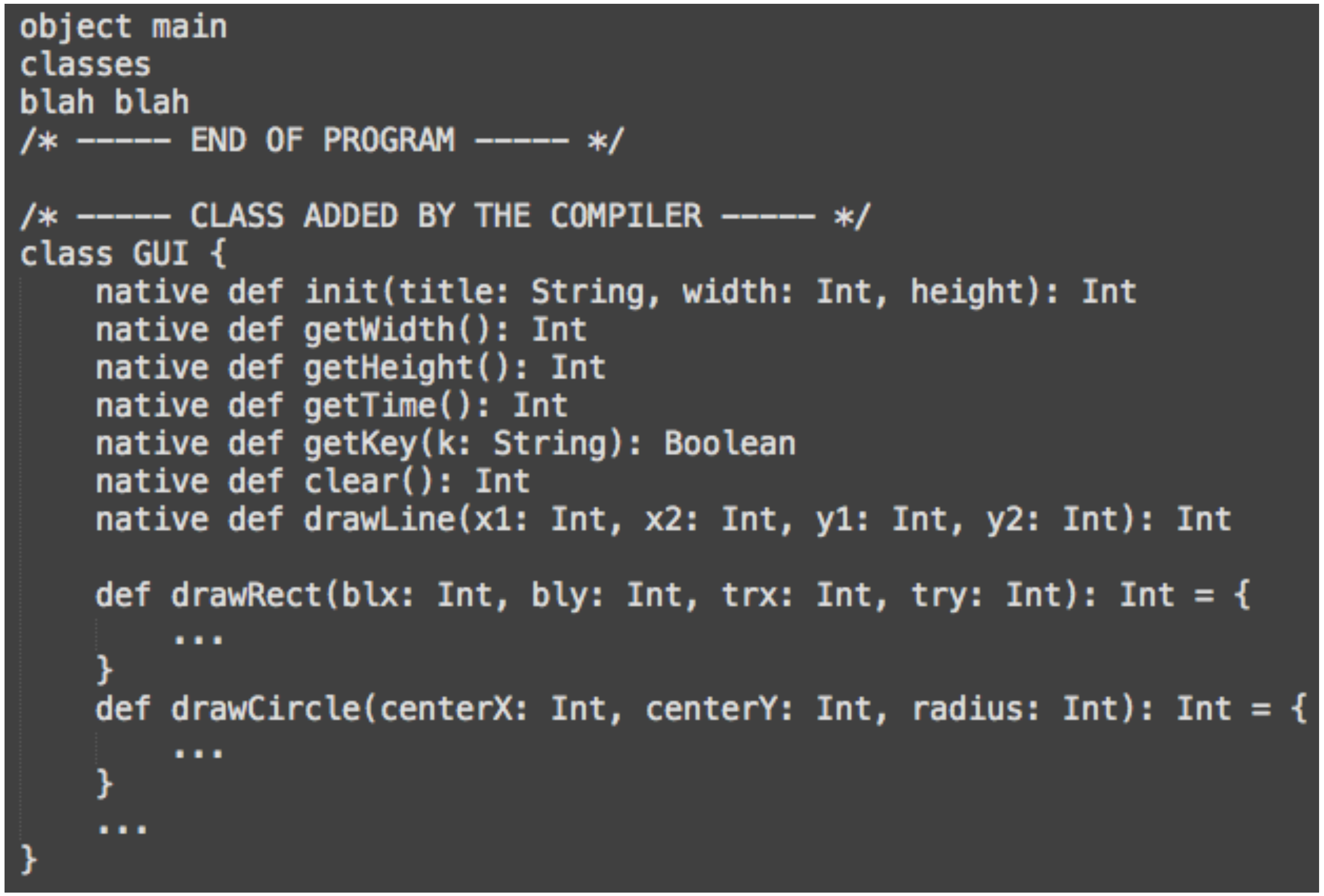
A lot of fun!

GENERAL IDEA - What the programmer sees

The GUI class has the following methods:

- getWidth, getHeight : Int -> get width and height of the GUI frame
- getTime : Int -> get time elapsed from the GUI creation
- getKey(k) : Boolean -> tell if a key is pressed
- clear -> clear the frame
- drawLine(x1, y1, x2, y2) -> draw a line on the frame

GENERAL IDEA - What Lexer and Parser see



```
object main
classes
blah blah
/* ----- END OF PROGRAM ----- */

/* ----- CLASS ADDED BY THE COMPILER ----- */
class GUI {
    native def init(title: String, width: Int, height): Int
    native def getWidth(): Int
    native def getHeight(): Int
    native def getTime(): Int
    native def getKey(k: String): Boolean
    native def clear(): Int
    native def drawLine(x1: Int, x2: Int, y1: Int, y2: Int): Int

    def drawRect(blx: Int, bly: Int, trx: Int, try: Int): Int = {
        ...
    }
    def drawCircle(centerX: Int, centerY: Int, radius: Int): Int = {
        ...
    }
    ...
}
```

The source code and a GUI class with:

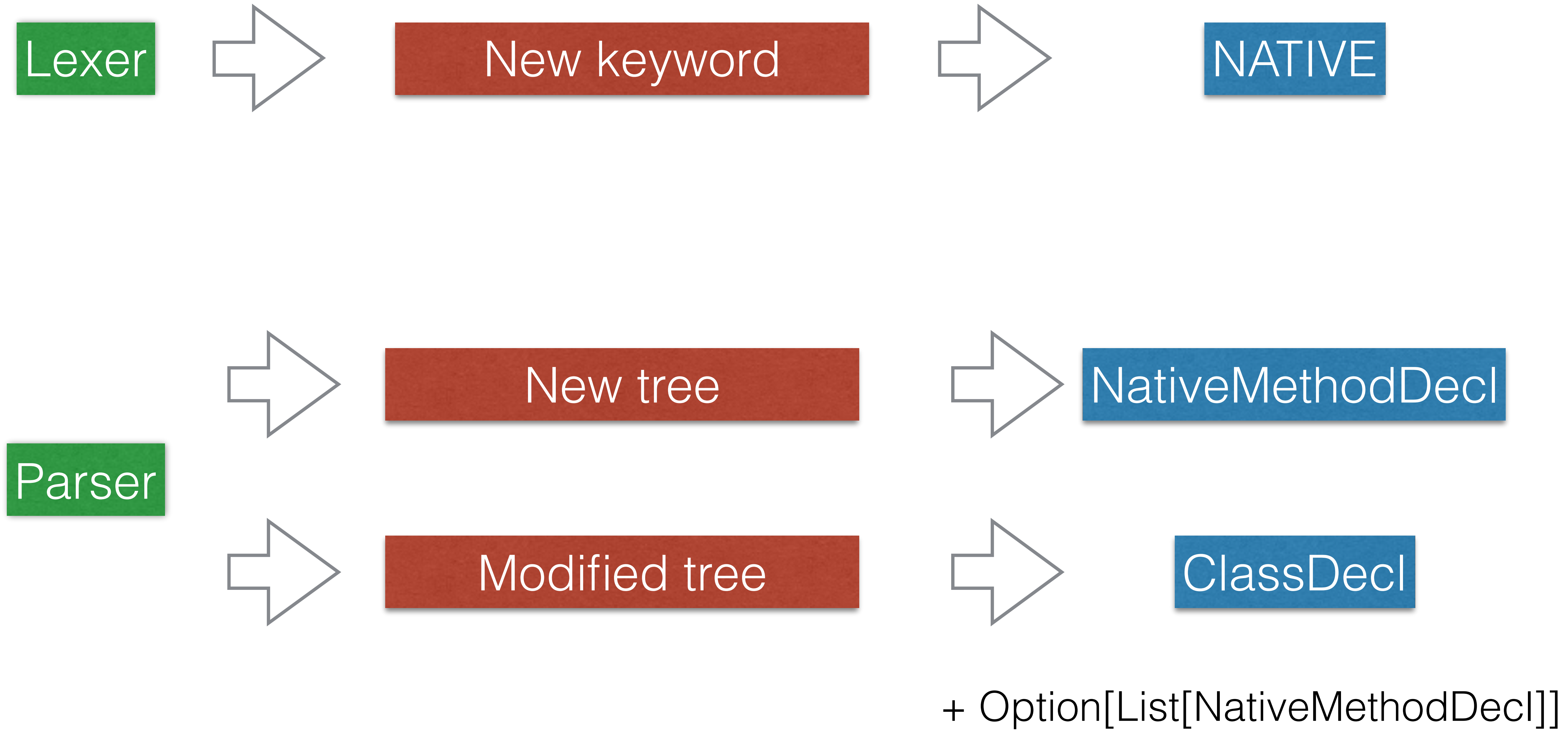
- native methods
- normal methods

GENERAL IDEA - What Lexer and Parser see

Native methods are not implemented in Tool, but are implemented later in bytecode in the code generation phase, calling the methods of the awt and Swing Java packages.

Non native methods are implemented in Tool using native methods, and are meant to be more high-level.

GENERAL IDEA - What Lexer and Parser see



GENERAL IDEA - What name analysis and type checking do

Nothing new, basically.

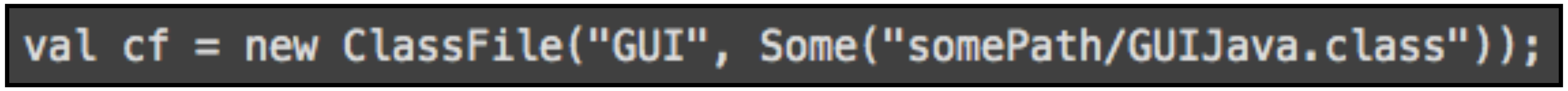
They assign symbols and types to native methods as they are normal ones.

GENERAL IDEA - What code generation does

Everything seems fine, but there's a problem:
the methods of the GUI class acts
on the frame created with init()…
but we don't have a way to keep a reference to
a frame in Tool!

GENERAL IDEA - What code generation does

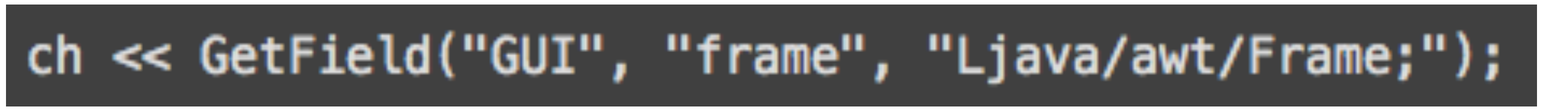
Solution:
the Tool “GUI” class extends a Java “GUIJava” class that has the frame as a field.



```
val cf = new ClassFile("GUI", Some("somePath/GUIJava.class"));
```

The GUIJava.class file will be somewhere in the compiler directory.

We’ll then be able to access the frame in a way like this:



```
ch << GetField("GUI", "frame", "Ljava/awt/Frame;");
```

Thank you