

# Introduction to Scala

It's pronounced *skah-lah*, not *scale-uh*.

# Scala History

- ▶ First release in 2003
- ▶ Designed by Martin Odersky
  - ▶ Student of ACM Turing Award winner Niklaus Wirth, creator of Pascal, Modula-2 and Oberon
  - ▶ With Philip Wadler, designer of GJ (Generic Java)
  - ▶ Chief implementer of Java 5 compiler still used today

Chief design goal: fuse object-oriented and statically-typed functional programming

# Notable Features of Scala

- ▶ Truly object-oriented
  - ▶ Everything is an object. Try`1.equals(1)`
  - ▶ No primitive/reference dichotomy
  - ▶ No static/instance context dichotomy
- ▶ Functional
  - ▶ First-class function values
  - ▶ Real lambda expressions
  - ▶ Immutable data
- ▶ Concise, pleasant syntax (~1/3 equivalent Java code)
- ▶ Many advanced features
  - ▶ Rich static typing with type inference
  - ▶ By-value parameters
  - ▶ Flexible syntax
  - ▶ Implicits
- ▶ JVM language with excellent Java interoperability
- ▶ “Scalable Language”
  - ▶ REPL, small scripts, million-LOC systems
  - ▶ Single machine or clusters
  - ▶ Start with simpler features and work

# Running Scala

- ▶ Install Java 8+ and Scala for system-wide use using the instructions linked in hw0
- ▶ Many ways to run Scala:
  - ▶ REPL
  - ▶ Command-line scripts
  - ▶ Command-line sbt projects
  - ▶ IntelliJ (or other IDE) projects
  - ▶ IntelliJ (or other IDE) sbt projects
  - ▶ IntelliJ (or other IDE) worksheets

# The Scala REPL

- ▶ Read-Eval-Print-Loop
- ▶ Enter an expression, REPL evaluates expression and prints its value
- ▶ Great way to become familiar with a language and try out parts of a project

```
1 $ scala
2 Welcome to Scala 2.12.8 (OpenJDK 64-Bit Server VM, Java 11.0.1).
3 Type in expressions for evaluation. Or try :help.
4
5 scala> 1 + 1
6 res0: Int = 2
7
8 scala> :quit
9 $
```

# Notable REPL Features

When entering blocks of code, class or function definition, more convenient to use :`paste`

```
1  scala> :paste
2  // Entering paste mode (ctrl-D to finish)
3
4  def max(x: Int, y: Int): Int =
5    if (x > y) x
6    else y
7
8  // Exiting paste mode, now interpreting.
9
10 max: (x: Int, y: Int)Int
11
12 scala>
```

Can run a file containing Scala code with :`load`

# Scala Scripts

On Unix, put this at the top of a text file:

```
1 #!/bin/sh
2 exec scala -savecompiled "$0" "$@"
3 !#
```

On Windows, put this at the top of a text file ending in `.bat`:

```
1 ::#!  
2 @echo off  
3 call scala -savecompiled %0 %*  
4 goto :eof  
5 ::!#
```

- ▶ `-savecompiled` saves a compiled version of your program with a `.jar` extension so that future execution is faster
- ▶ The other two args tell the shell to run the Scala program and pass it the command line args

Run with:

```
1 $ scala myscript.scala
```

# IntelliJ Scala Worksheets

A Scala worksheet is a text file containing Scala code whose name ends in `.sc` (also works in Eclipse). Run with the play button in upper left corner

The screenshot shows the IntelliJ IDEA interface with a Scala worksheet open. The title bar says "values-variables.sc". The code editor contains the following Scala code:

```
// Literals and basic types as in Java, but with different names.
// Variable definitions introduced with var or val
// A var is reassignable. Note type annotation syntax.
var x: Int = 1
x = 2

// A val is not reassignable, like a final variable in Java.
// Note type inference -- no type annotation, but still static type
val y = 3.14 // Type is Double, inferred by literal
// This would not compile if uncommented because y is a val:
// y = 6.28

// Blocks are sequences of expressions enclosed in {}
// Last expression is value of the block.
// Also notice the static lexical scoping --
// The y inside the block shadows the y above
val s = {
    val y = 1 //
    y == 1 // These two lines will generate a warning that they are pure expressions
    2 // in statement position, i.e., they don't produce the value of the block,
    // "buckle my shoe"
}

// What's the type and value of y at this point?
val areSame: Boolean = x == y
val favorite = "Tame Impala"

// Triple-quoted String. stripMargin method removes leading spaces before |
val letItHappen =
  """All this running around
   |Trying to cover my shadow
   |A notion growing inside
   |Now all the others seem shallow
   |All this running around
   |Bearing down on my shoulders
   |I can hear an alarm
   |It must be morning"""
  .stripMargin

val music = s"Play my favorite band, $favorite"
val have = s"Is my favorite band Tame Impala? Result: ${favorite == "Tame Impala"}"
```

The code editor has several tabs: "code", "basics", and "values-variables.sc". The status bar at the bottom shows "Push successful: Pushed 1 commit to origin/master (today 14:51)". The bottom right corner features the Georgia Tech logo.

# Command-Line sbt Projects

- ▶ Create a directory for your project
- ▶ In your project's directory, create a `build.sbt` with minimal contents:

```
1 ThisBuild / scalaVersion := "2.12.8"
2 ThisBuild / organization := "edu.gatech.cs2340"
3
4 lazy val hello = (project in file("."))
5   .settings(
6     name := "YOUR NAME HERE"
7   )
```

Launch sbt at the command line, which will download some things, create a target directory and a `project` directory with `build.properties` (yes, there's a configuration for the build tool)

```
1 $ sbt
2 [info] Loading global plugins from /Users/chris/.sbt/1.0/plugins
3 [info] Loading project definition from
4   /Users/chris/scratch/sample/project
5 [info] Updating
6   ProjectRef(uri("file:/Users/chris/scratch/sample/project/"),
7   "sample-build")...
8 [info] Done updating.
```

## sbt Basics

- ▶ sbt is interactive. Use help to list commands
- ▶ Common tasks:
  - ▶ `compile`
  - ▶ `test`
  - ▶ `run` – will find and list all objects with main methods
- ▶ Assumes Maven standard source tree structure and its own configuration files:

```
1 build.sbt
2 project
3   build.properties
4 src
5     main
6       java
7       scala
8     test
9       java
10      scala
11
```

## IntelliJ sbt Projects

Can also use IntelliJ's New Project wizard to create a Scala sbt project.