# CSE 250 - Data Structures: Background Review

Andrew Hughes

SUNY at Buffalo
Computer Science and Engineering

Fall 2019

## Outline

## Logistics

- My office: 348 Davis Hall
  - Office hours calendar will be posted to Piazza.
  - Recitations start week 2 – treated similar to office hours.
  - No lecture on Monday.
- Make sure you have access to Piazza (`https://piazza.com/buffalo/fall2019/cse250/`)
  - Send me email (ahughes6@buffalo.edu) if you need access.
  - Always begin subject line with **[CSE 250]** for any course email.
  - Emails not from your @buffalo.edu address will be deleted/ignored.
- Access to AutoLab will start after first assignment (`https://autograder.cse.buffalo.edu/`)
  - First assignment will be posted end of this week/early next week.

## Outline

## IntelliJ Recommended

- Use a comfortable environment.
  - IDEs: IntelliJ (recommended/supported), Eclipse, . . .
  - Textual environments: VIM, EMACS, Sublime, Atom, . . .
- We provide projects generated by IntelliJ for each assignment.
  - Ensure you produce the appropriate file(s) for submission.
  - Must build/run on Autolab, regardless of your production route.
- We are using Scala 2.13 for this semester.
- We will use ScalaTest for testing purposes.

## Hello, World!

```scala
object HelloWorld {
    def main(args: Array[String]): Unit = {
        println("Hello, World!")
    }
}
```

Scaladoc will be your friend:
https://www.scala-lang.org/api/current/index.html

## Coding Style

```
def doThings() = {
              val ILikeLlamas = 10
val PeachesAreGreat = for (i <- 1 to 5) yield i

 val QQ = PeachesAreGreat.map(_+ILikeLlamas)

    // This is a for loop.
                    for (q <- QQ) println(q)
                    // This is a loop with a 4.
    for (i <- 0 until 4) println(i)
    5
}
```

- A note on code style:
  - Use proper indentation and whitespace.
  - Include comments that follow the flow of your ideas.
  - Use variable names that make their purpose known.
- Don't expect course staff to understand code with nonsense variable names and no comments.

## Utilizing Office Hours

### How to obtain assistance and succeed

- Draw diagrams of your ideas.
- Write pseudocode explaining your approach.
    - !!!Do this before writing code!!!
- Follow code style guidelines.
- Explain approaches/tests you have tried that failed to obtain the expected outcome.
    - Ask about ideas on how to test your code.

# Fundamental Types

Scala has the usual types you would expect.

| | |
|---|---|
| `Boolean` | Boolean value, `false` or `true` |
| `Char` | 16-bit unsigned integer |
| `Byte` | 8-bit signed integer |
| `Short` | 16-bit signed integer |
| `Int` | 32-bit signed integer |
| `Long` | 64-bit signed integer |
| `Float` | single-precision floating-point number |
| `Double` | double-precision floating-point number |
| `Unit` | no value – declared by `()` |

Scala has no primitives until there are... until there aren't.

## Expressions

Every expression has a type.

- Can explicitly declare type or allow Scala to infer.
- Be aware of the relation between types.
  - Why is the cast to `Float` necessary?

```
val x: Float = (5 / 2.0).asInstanceOf[Float]

val holder = 15 + 10.2 * 9.3f

def lotsOfFun(x: Int) = "fun" * x
```

Examples/HelloWorldAtLarge.scala

## Expressions

**Keep types consistent!**

```
val res = if (x > 0) "Positive" else -1

val better = if (x > 0) "Positive" else -1.toString()
```

Examples/HelloWorldAtLarge.scala

- What happens when two types are possible?
  - `Any` or `AnyRef` is used – bad practice.

## Expressions

Each block has a type.

```
def doThings() = {
  val ILikeLlamas = 10
  val PeachesAreGreat = for (i <- 1 to 5) yield i

  val QQ = PeachesAreGreat.map(_ + ILikeLlamas)

  // This is a for loop.
  for (q <- QQ) println(q)
  // This is a loop with a 4.
  for (i <- 0 until 4) println(i)
  5
}
```

Examples/HelloWorldAtLarge.scala

- Be careful not to omit the equals (=) when declaring functions!
  - `def` without = assumes the type `Unit`.

## Expressions

Blocks can also be used for assignments.

```scala
val blockAssign = { val x = 10; val y = 20; (x, y) }
val butterBlock = {
  val pastry = "croissant"
  val flavor = "PB&J"
  flavor + ' ' + pastry
}
```

Examples/HelloWorldAtLarge.scala

- Semi-colons needed for multiple expressions on one line.

## Definitions

**Definition**

Something that can be changed is **mutable**.

**Definition**

Something that cannot be changed is **immutable**.

- Scala differentiates between this.
  - `val` – **val**ue that cannot be reassigned.
  - `var` – **var**iable that can be reassigned.

## Val vs Var

```
scala> val s = mutable.Set(1,2,3)
s: scala.collection.mutable.Set[Int] = HashSet(1, 2, 3)

scala> s += 4
res0: s.type = HashSet(1, 2, 3, 4)
```

Why can we reassign here?

## Class vs Object vs Trait

We won't worry too much about the differences for now.

| | | |
|---|---|---|
| `class` | – | normal OOP class. |
| `object` | – | similar to `class` but only one instance may exist. |
| `trait` | – | also similar, but cannot be instantiated. |
| `case class` | – | similar to class, provides special functionality. |

Only allowed one super class, but any number of traits is allowed.

## Class vs Object vs Trait

Companion objects can define an `apply` method to avoid `new`.

```
scala> :paste
class Register(val x : Int) {
  def addValue(y: Int) = x + y
}
object Register {
  def apply(x:Int) = new Register(x)
}
scala> val reg5 = new Register(5)
reg5: Register = Register@146f3d22
scala> val reg10 = Register(10)
reg10: Register = Register@43b172e3
scala> print(reg5.addValue(100))
105
scala> print(reg10.addValue(100))
110
```

## Outline

## The REPL

If you don't how code behaves, use the REPL environment to quickly test it.

- Can be entered in IntelliJ using `Ctrl+Shift+D`.
- Once Scala console is open:
  - Highlight line and press `Ctrl+Shift+X` to execute.
  - Copy+paste line into console and press `Ctrl+Enter`.

`:paste` mode can be used to enter a block of code.

## ScalaTest

There is a rich testing environment provided by `ScalaTest`.

- Tests need not be complicated or complete at the start of development.
- Helpful for detecting problems later on.

## ScalaTest

Describe tests in English.

```
class HelloWorldTest extends FlatSpec {
 "HelloWorld.doThings()" should "return 5" in {
    assert(HelloWorld.doThings() == 5)
 }
 it should "not return 10" in {
  assert(HelloWorld.doThings() != 10)
 }
 "HelloWorld.x" should "have type Float" in {
  assert(HelloWorld.x.isInstanceOf[Float])
 }
 "Register(0).addToValue" should "return the input value"
     in {
  val reg = Register(0)
  for (i <- 1 to 10000) assert(reg.addToValue(i) == i)
 }
}
```

Examples/HelloWorldTest.scala

## Outline

## Questions?

## Bibliography

C. S. Horstmann, *Scala for the Impatient*.
Addison-Wesley, 2017.