# Scala Values and Variables

Georgia Tech

# Values and Variables

- Like Java, every Scala variable has a name and a type
- Variable definitions start with `var` or `val`.
  - `var`s are reassignable
  - `val`s like Java's `final`

```
1  var x: Int = 1
2  x = 2
3
4  val y = 3.14 // Type is Double, inferred by literal
5  y = 6.28 // Won't compile -- y is a val
```

Georgia Tech

# Basic Types

Same basic types as Java, but different names:

- Byte, Short, Int, Long, Char
- String
- Float
- Double
- Boolean

Literals same as in Java

Georgia Tech

# Blocks

Blocks are enclosed in curly braces. Last expression gives value of the block.

```
1  val s = {
2    1
3    2
4    "buckle my shoe"
5  }
```

Value of `s` above is `"buckle my shoe"`

Georgia Tech

# Basic Operators

- Basic arithmetic, logical, relational and bitwise operators like Java's
- All operators are actually methods (more later)
- Precedence based on first character of operator:
  - (all other special characters) then *,/,% then +,- then : then =,! then <,> then & then ^ then | then (all letters) then (all assignment operators)
- Associativity based on last character of operator
  - Operators ending in : invoked on right operand
  - All others invoked on left operand

Georgia Tech

# Object Equality

- All objects have equals methods, just like Java but the equality operators are different
  - == same as equals method
  - `eq` is alias testing operator
- We'll discuss implementation of equals and hashCode in a future lecture.

Georgia Tech

# Basic Sequences

## Lists are immutable Sequences of like-typed elements

```
1  val xs: List[Int] = List(1, 2, 3)
2  xs(0) = 42 // Won't compile
3
4  // Add elements to head of list with cons operator, ::
5  val ys = 0::xs
6  ys == List(0, 1, 2, 3)
7
8  // Cons returns a new list
9  xs != ys
10
11 // To "modify" xs, reassign (only works if xs is a var)
12 xs = 0::xs
```

## Arrays are mutable fixed-sized Sequences of like-typed elements

```
1  val zs: Array[Int] = Array(1, 2, 3)
2  zs(0) = 42
3  zs == Array(42, 2, 3)
```

Georgia Tech

# Sets and Maps

Sets are immutable by default, so we "add" to them with reassignment

```
1 var trooperSet = Set("Thorny", "Farva", "Mac", "Mac")
2 trooperSet == Set("Thorny", "Farva", "Mac")
3 trooperSet += "Rabbit"
4 trooperSet.contains("Rabbit")
```

Map elements created with 2-tuples, which are usually created with `->`

```
1 var majors = Map(
2   ("CS", "Computer Science"),
3   "CM" -> "Computational Media",
4   "EE" -> "Electrical Engineering"
5 )
6 majors += "IE" -> "Industrial Engineering"
7 majors("IE")
8 majors.getOrElse("AA", "Unknown Major")
```

`->` uses implicit conversion to create `Tuple2` instances.

Georgia Tech

# Conclusion

- In Scala, every value is an object, that is, an instance of a class.
  - Scala compiler makes basic types as efficient as in Java while providing the elegance of the uniform "everything is an object" abstraction
- Scala is statically typed but performs type inference to make simple REPL interactions or scripts as convenient as dynamically-typed langauges like Python

Georgia Tech