Objects Everywhere

Pure Object Orientation

- A pure object-oriented language is one in which every value is an object
- If the language is based on classes, this means that the type of each value is a class
- A class such as *Int* or *Boolean* is represented by the computer quite differently from an object
- An object is typically a multi-word record on the heap and an *Integer* or a *Boolean* is just a primitive value that can sit in a register
- Conceptually, types such as *Int* or *Boolean* do not receive any special treatment in Scala
- They are like the other classes, defined in the package scala
- For reasons of efficiency, the Scala compiler represents values of type *scala.Int* by 32-bit integers and values of type *scala.Boolean* by Java's Booleans, whereas, a normal object would be represented as some form of record with multiple fields in the heap of the program execution

Pure Booleans

- The Boolean type maps to the JVM's primitive type Boolean
- But one could define it as a class from first principles:

```
abstract class Boolean extends AnyVal:
  def ifThenElse[T](t: => T, e: => T): T
  def && (x: => Boolean): Boolean = ifThenElse(x, false)
  def || (x: => Boolean): Boolean = ifThenElse(true, x)
  def unary_!: Boolean = ifThenElse(false, true)
  def == (x: Boolean): Boolean = ifThenElse(x, x.unary_!)
  def != (x: Boolean): Boolean = ifThenElse(x.unary_!, x)
```

- Boolean Constants:

```
object true extends Boolean:
  def ifThenElse[T](t: => T, e: => T) = t

object false extends Boolean:
  def ifThenElse[T](t: => T, e: => T) = e
```

Exercise: *Provide an implementation of an implication operator ==> for the class written above.*

a ==> b <=> b || !a

```
extension (x: Boolean):
  def ==> (y: Boolean): Boolean = x.ifThenElse(y, true)
```

The Class Int

- Here is a partial specification of the class *scala.Int*:

```
class Int:
  def + (that: Double): Double // same for -, *, /, %
  def + (that: Float): Float
  def + (that: Long): Long
  def + (that: Int): Int

  def << (cnt: Int): Int // same for >>, >>>

  def & (that: Long): Long // same for |, ^
  def & (that: Int): Int

  def == (that: Double): Boolean // same for !=, >=, <=, >, <
  def == (that: Float): Boolean
  def == (that: Long): Boolean
  def == (that: Int): Boolean
```