Lecture 2 Jan 16

Int, Boolean, Byte, Char, Double, Short, Float

But, in Java int V.S. Integer

uniform → no "primitive"
- everything is an object
- every operation is a function call

```
scala> 8.toString()
res0: String = 8
```

- function call does not change input
- Strings, Arrays, Lists, Sets ...
  → in Scala "rich" types
- There's no ++ or -- in Scala.
  Semi-colons are inferred.

```
val x = 3
       +4  ;
```

- return is not necessary.

```
def foo (x : Int) : Int = {
x+1
}
```

```
def foo (x :Int) = x + 1
```

- Conditionals :
  if statements
  → In scala, it gets evaluated and

if statements
→ In scala, it gets evaluated and returns something.
→ They are expressions.

```
scala> val x = if (3>4) 17 else 25
x: Int = 25
```

→ Subsumes tenary operator
In Java:

```
int x = (3>4) ? 17 : 25 ;
```

```
scala> if(3<4) { if (2<3) 7 else 9} else 88
res0 : Int = 7
```

```
x = { ____
      ____
      ____     // Last Line to be returned
    }          // to x.
```

Loops
While, do ......

```
while (     ) { ____
                ____
                ____
              }
// return type is Unit
```

Java: for(.......) {   }
In Scala, no X!
→ We don't want things updated.
• for - loops → iterator

→ We don't want things updated.

- for-loops → iterator

```
scala> for (i <- 0 to 10) println(i)
```

→ Like creating 11 objects, and iterating them all.
→ Not updating the value of same i

Syntax : for (var <- expr) body

- for-loops → comprehensions
  → new collections from old

```
scala> for ( c <- "space") yield (c+1).toChar
res0: String = ...
```

r

<u>Arrays</u>

val x = new Array[Int](5)
→ Int : initilizes / with zeroes
→ String : as null.

- ArrayBuffer : can update size