19CSE313 – PRINCIPLES OF PROGRAMMING LANGUAGES

Basic Types and Operations

SOME BASIC TYPES

Basic Type	Range
Byte	8-bit signed two's complement integer (-27 to 27 - 1, inclusive)
Short	16-bit signed two's complement integer (-215 to 215 - 1, inclusive)
Int	32-bit signed two's complement integer (-231 to 231 - 1, inclusive)
Long	64-bit signed two's complement integer (-263 to 263 - 1, inclusive)
Char	16-bit unsigned Unicode character (0 to 216 - 1, inclusive)
String	a sequence of Chars **(java.lang)**
Float	32-bit IEEE 754 single-precision float
Double	64-bit IEEE 754 double-precision float
Boolean	true or false

- **Note: Collectively, types Byte, Short, Int, Long, and Char are called *integral types*.**
- **The integral types plus Float and Double are called *numeric types*.**
- **All types other than string are members of scala package**

LITERALS (CONSTANTS)

- All of the basic types listed in Table 5.1 can be written with *literals*
- **Integer literals:**
  - Integer literals for the types Int, Long, Short, and Byte come in two forms: decimal and hexadecimal.
  - The way an integer literal begins indicates the base of the number.
  - If the number begins with a 0x or 0X, it is hexadecimal (base 16), and may contain 0 through 9 as well as upper or lowercase digits A through F.
  - Some examples are:

```
scala> val hex = 0x5
hex: Int = 5
scala> val hex2 = 0x00FF
hex2: Int = 255
scala> val magic = 0xcafebabe
magic: Int = -889275714
```

- Note that the Scala shell always prints integer values in base 10, no matter what literal form you may have used to initialize it.
- Thus the interpreter displays the value of the hex2 variable you initialized with literal 0x00FF as decimal 255.

DECIMAL LITERALS

- If the number begins with a non-zero digit, and is otherwise undecorated, it is decimal (base 10).

  For example:

```
scala> val dec1 = 31

dec1: Int = 31

scala> val dec2 = 255

dec2: Int = 255

scala> val dec3 = 20

dec3: Int = 20
```

LONG LITERALS

- If an integer literal ends in an L or l, it is a Long; otherwise it is an Int.

  Example:

```
scala> val prog = 0XCAFEBABEL
prog: Long = 3405691582

scala> val tower = 35L
tower: Long = 35

scala> val of = 31l
of: Long = 31
```

SHORT OR BYTE

- If an Int literal is assigned to a variable of type Short or Byte, the literal is treated as if it were a Short or Byte type so long as the literal value is within the valid range for that type.

  For example:

```
scala> val little: Short = 367
little: Short = 367

scala> val littler: Byte = 38
littler: Byte = 38
```

FLOATING POINT LITERALS

- Floating point literals are made up of decimal digits, optionally containing a decimal point, and optionally followed by an E or e and an exponent.

Example:

```
scala> val big = 1.2345

big: Double = 1.2345


scala> val bigger = 1.2345e1

bigger: Double = 12.345


scala> val biggerStill = 123E45

biggerStill: Double = 1.23E47
```

- Note that the exponent portion means the power of 10 by which the other portion is multiplied. Thus, 1.2345e1 is 1.2345 *times* 101, which is 12.345.

FLOAT LITERALS

- If a floating-point literal ends in an F or f, it is a Float; otherwise it is a Double.
- Optionally, a Double floating-point literal can end in D or d.

Some examples of Float literals:

```
scala> val little = 1.2345F
little: Float = 1.2345

scala> val littleBigger = 3e5f
littleBigger: Float = 300000.0
```

That last value expressed as a Double could take these (and other) forms:

```
scala> val anotherDouble = 3e5
anotherDouble: Double = 300000.0

scala> val yetAnother = 3e5D
yetAnother: Double = 300000.0
```

CHARACTER LITERALS

- Character literals are composed of any Unicode character between single quotes, such as:

```
scala> val a = 'A'

a: Char = A
```


- Character literals are composed of any Unicode character between single quotes, such as:

```
scala> val a = 'A'

a: Char = A
```


- In fact, such Unicode characters can appear anywhere in a Scala program. For instance you could also write an identifier like this:

```
scala> val B\u0041\u0044 = 1

BAD: Int = 1
```

ESCAPE SEQUENCES

Literal	Meaning
\n	line feed (\u000A)
\b	backspace (\u0008)
\t	tab (\u0009)
\f	form feed (\u000C)
\r	carriage return (\u000D)
\"	double quote (\u0022)
\'	single quote (\u0027)
\\	backslash (\u005C)

Example:

```
scala> val backslash = '\\'
backslash: Char = \
```

STRING LITERALS

- A string literal is composed of characters surrounded by double quotes:

```
scala> val hello = "hello"
hello: String = hello
```

- The syntax of the characters within the quotes is the same as with character literals.

For example:

```
scala> val escapes = "\\\"\'"
escapes: String = \"'
```

BOOLEAN LITERALS

- The Boolean type has two literals, true and false:

```
scala> val bool = true
bool: Boolean = true

scala> val fool = false
fool: Boolean = false
```

STRING INTERPOLATION

- Allows you to embed expressions within string literals
- Provides a concise and readable alternative to string concatenation

```
scala> val name = "reader"
val name: String = reader
scala> println(s"Hello, $name!")
Hello, reader!


scala> s"The answer is ${6 * 7}."
val res2: String = The answer is 42.
```

OPERATORS

- Operators are methods
- For example, 1 + 2really means the same thing as 1.+(2).
- In other words, class Int contains a method named +that takes an Int and returns an Int result.
- This + method is invoked when you add two Ints:

```
scala> val sum = 1 + 2 // Scala invokes 1.+(2)

sum: Int = 3

scala> val sumMore = 1.+(2)

sumMore: Int = 3

scala> val longSum = 1 + 2L // Scala invokes 1.+(2L) – overloaded version

longSum: Long = 3
```

TYPES OF OPERATORS

Type	Symbol
Unary	-
Arithmetic	+, -, *, / , %
Relational	>, <, >=, <=, !
Logical	&&, ||, !, (&,| to evaluate R.H.S)
Bitwise	&,	, ^, ~ (complement)
Shift	<<, >>, >>> (unsigned shift right)
Object Equality	==, !=

OPERATOR PRECEDENCE

Operator	Precedence
* / %	1
+ -	2
:	3
= !	4
< >	5
&	6
^	7
|	8
All letters	9
All operators	10

RICH WRAPPER CLASSES AND OPERATIONS

Code	Result
0 max 5	5
0 min 5	0
-2.7 abs	2.7
-2.7 round	-3L
1.5 isInfinity	false
(1.0 / 0) isInfinity	true
4 to 6	Range(4, 5, 6)
"bob" capitalize	"Bob"
"robert" drop 2	"bert"

Basic type	Rich wrapper
Byte	scala.runtime.RichByte
Short	scala.runtime.RichShort
Int	scala.runtime.RichInt
Long	scala.runtime.RichLong
Char	scala.runtime.RichChar
Float	scala.runtime.RichFloat
Double	scala.runtime.RichDouble
Boolean	scala.runtime.RichBoolean
String	scala.collection.immutable.StringOps

THANK YOU