1-21-2011
Opening Discussion

- Minute Essay comments
  - Copy/paste with middle click on Linux. On Windows?
  - vi color-coding.
  - Never touch the mouse.
  - Low learning curve in Scala.
  - Case sensitivity
  - Going to the web for help. Google
Key Terms

- **Token** – A set of characters that has meaning to the language. (Words)
- **Statement** – A set of tokens that give a complete instruction. (Sentences)
- **Expression** – Tokens put together that produce a value. (Phrases)
- **Type** – All values have types. A type is a set of values and the operations allowed on them.
- **Literal** – A token that represents a value.
  - Numeric, String, Character, Boolean
In Scala, as with most programming languages, programs are made by putting together statements.

In Scala, any expression is a valid statement as are a few other constructs.

Statements end with semicolons, but they will be inferred at the end of a line if they make sense so you rarely type them.
We can build longer expressions by putting literals together with operators.

Let's start off by playing with some of the numeric operations you are probably familiar with.

- +, -, *, /

You can get the remainder after division with %.
An object is defined to be information along with the things you can do with that information.

The information in an object is called the properties.

The actions are called methods.

In Scala, even things like `Int` are objects and have methods on them.
Methods

- The normal way to call a method in Scala (and most other object-oriented languages) is to put a period after the object and follow it with the method name.
- The REPL will do tab completion and list methods for you.
- Let's look at the methods on some basic types and try calling them.
Arguments

- Some methods need additional information to work.
- To give this to the method we pass in arguments.
- Arguments are put in parentheses and separated by commas if there is more than one.
- The parentheses are generally optional in Scala if there is no argument.
All the “operators” in Scala are really just methods.

Scala allows any method with zero or one arguments to be called with an operator syntax.

That means you leave off the dot and the parentheses.

If a method takes no arguments you can call it without the dot.
Bases and Binary

- The decimal numbers we use are base 10. Each digit to the left is a higher power of 10.

- There is nothing special with decimal (other than perhaps we have 10 fingers). Other bases are equally valid.

- Computers use binary numbers to store everything.

- All digits are 0 or 1 and each position is a higher power of 2.

- toBinaryString
Adding binary numbers is very easy. Just do the long addition that you are used to.

You will carry a lot more frequently because anything above 1 causes a carry.

Let's run through some examples.

Consider implications of fixed precision.
Negative Numbers

- We don't have a – in the computer for negative numbers. All we have are 1 and 0. So how do we make negative numbers?
- Remember the definition of negative numbers as additive inverse.
  - a+(-a)=0
- We want to preserve this to keep addition simple.
- This gives us 2s-compliment numbers.
Binary Multiplication

- Multiplying binary numbers works just like long multiplication with decimals, but easier.
- My only recommendation is you only add two numbers at a time and take it in steps.
Hexadecimal

- Binary is unwieldy for humans because of the large number of digits.
- Hexadecimal (base 16) is commonly used because it converts nicely to binary, but has few digits.
- Four bits is a hex digit. Start at the right and group bits by 4.
- Use letters A-F for numbers 10-15.
- Hex literals start with 0x
- toHexString
Octal

- Octal (base 8) is less common than hex, but not uncommon.
- Group bits into groups of three.
- Octal literals and toOctalString().
The math Object

- For other math functions use methods on the math object.
- For example, use `math.sqrt()` to take the square root of a number.
Convert 276 to binary.

We have our first quiz next class.