Scala Expressions and Types

9-1-2010
Opening Discussion

- Let's look at solutions to the interclass problem.
- Command line impressions.
  - Mix of easy and complex.
  - Has a learning curve.
Programs are typically written as plain text files and should be edited with a text editor.

Notepad is a basic text editor on Windows.

Word is NOT a text editor.

Some text editors are better than others for programming.

In this class we will use vi.
The vi editor is standard on Linux which is one reason we like to use it.

It is also good for programming.

Has modes. Start in command mode. You type in an edit mode.

- i – insert
- I – insert at beginning of line
- a – append
- A – append at end of line
- R – replace characters
Other Commands

- `x` – delete characters
- `dd` – delete lines
- `yy` – yank/copy lines
- `p` or `P` – paste before or after
- `r` – replace a single character
- `J` – join lines
- `/` and `n` – search for something and next
- `cw` – change a word
- `.` - repeat last command
- `u` and `Ctrl-r` – Undo and redo
Let's make a directory and use vi to write our first Scala program then run it.

The standard first program is “Hello World.” and I don't want to break with tradition.
Scala REPL

- If you just type scala and don't provide a file name to run, it will drop you into the RELP (Read, Evaluate, Print Loop).
- Here you can enter individual commands and have them run.
- It is great for testing things out and getting to learn the language.
Key Terms

- **Token** – A set of characters that has meaning to the language.
- **Expression** – One or more tokens put together that produce a value.
- **Type** – All values have types. A type is defined as a set of values and the operations that are 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 %.

Play with semicolon inference a bit.
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.
What in today's lectures was confusing? Was there anything that surprised you?

I'm not going to start grading interclass problems until the 13th. However, I highly recommend you do them anyway.

Interclass Problem: Play with the Scala REPL for a bit. Use it like a calculator and find instances where the answers are unexpected.