I always open class with a bit of a discussion.
Course web site:  
http://www.cs.trinity.edu/~mlewis/CSCI1321-F11/

Office: HAS 201K

Office hours: 10:00-3:00 T & 11:00-4:00 R.
  - Appointment, Piazza, Google+

Phone: 7022

E-mail: mlewis@trinity.edu (This is the best way to reach me most of the time.)

There is a schedule on the web site listing all topics and when things are due. There are also links to my lecture notes.
We will use “An Introduction to the Art of Programming using Scala”.

This is a book that I am in the process of writing. You can get to the PDF under the Links page.

You will be expected to do readings from this book and we will be following it closely.
Coding

- The largest chunk of your grade in this class is determined by the 4 assignments that you will do over the course of the semester. You will also be expected to complete 10 interclass problems.
- These are mostly small problems that you will solve by writing code, but some will link together allowing you to produce a larger product.
- You can work with others on design and get help on specific problems, but what you turn in for assignments must be your own construction. Everything you turn in for a grade is pledged if you are under the honor code. IcPs can be done with others.
Grading

- Your grade comes from five different components.
  - Assignments (4) – 40%
  - Tests (2) – 30%
  - Quizzes (6 with lowest dropped) – 10%
  - Interclass Problems – 10%
  - Class Participation – 10%

- The midterm and final are both equally weighted.
More on Grades

- The quiz questions are modeled after test questions so you will have an idea of the style to expect on the test.
- Class participation includes attendance and your actual participation during class.
Interclass Problems

- On the “Show Your Code” days I will call on roughly half the class “randomly” to present their answers.
- Each of you will be called on a total of five times with each time being worth two points to your final average.
- You should do all the IcPs.
An object is a grouping of data and functionality. We use the terminology of member data and methods.

Scala is a class-based OO language. Programmers define classes that are used as the blueprints of objects.

Code inside of classes looks much like that anywhere else in Scala.

Takes arguments and instantiated with new.
This semester we will be working pretty much exclusively in Eclipse so let's pull that up, make a project, and write a little code.

A bank account makes a nice simple first example.
Members of classes can be “hidden” so that only some code can see it.

Three visibilities:
- Public is the default.
- Private can only be seen by code in that class.
- Protected can be seen in this class and subclasses.

You can put a visibility on any declaration.
- `val`, `var`, `def`, `class`
- `type`, `object`, `trait` (you don't know these)
Arguments as Members

- By default the arguments to a class are not members.
- To make them so put either val or var in front of the name.
- You can also put a visibility in front of the val or var to make it private or protected.
Scala supports another declaration similar to a class called an object declaration.

This creates a singleton object in the scope that you declare it.

Making an object with the same name as a class creates a companion object. The companion object has access to the private members of the class, but can be accessed without instantiating the class.
Bigger Programs

- The REPL and scripts are good for small programs.
- OO really benefits large programs.
- Typically you put each class/object in its own file with the same name as the class/object.
- Compile using scalac.
- In an object you put a main that allows you to run the object with scala.
  - def main(args:Array[String]) { ... }
At the end of every class I will ask you a question and have you write a short response. Make sure you have your name on it as this is how I keep attendance. You can also use it for general questions/feedback.

- What are your thoughts on this class and the description?
- Is there anything in particular you want to make sure we do in this course?