Slide 1

Administrivia

- Homework 2 on the Web; due in a week. You should be able to do the first two problems with only the material from chapter 3, the others with material up through chapter 4.
- Quiz 1 next Tuesday. Open book, open notes, about ten minutes. More about possible topics next time.

Slide 2

Scala Basics — Review

- We've talked about some basic types of data you can work with in Scala, plus how to give names to things ("variables") how to do basic calculations, how to get input from a human using the program, and how to display results.
- Something not discussed explicitly in class: By default programs execute one statement at a time, in order — "sequential execution". (This can be important!)
- We used this to write a simple example program last time, to count out a number of cents in dollars, etc. A possible improvement would print only the nonzero values. How to do that?

Slide 3

Sidebar: "Write For Your Audience"

- In writing courses students are often told to know the intended audience and write for them.
- Something similar applies to programming, but there are several audiences:
  - The compiler. Cares about syntax and semantics, not so much about layout, names, comments, etc.
  - Human readers of the program. May not notice errors in syntax and logic but are helped by good variable names, comments, layout.
  - Human users of the program. Don't care about any of the above but do care about useful prompts and output.

Slide 4

Conditional Execution

- So far all our programs have executed the same statements every time, just maybe with different numbers.
- Often, though, we want to be able to do different things in different circumstances — for example, print an error message and stop if the input values don't make sense (such as a negative number for the program to count out change), or in the same program printing only nonzero results.
- So, Scala (like most languages) provides some constructs for *conditional execution*. Before we talk about them, we need . . .

Slide 5

Boolean Expressions

- A *Boolean value* is either *true* or *false*; a *Boolean expression* is something that evaluates to true or false.
- We can make simple examples in Scala using familiar math comparison operators (except that the ones for which the keyboard doesn't have a symbol require more than one character). Examples:
  - `x > 10`
  - `y <= 5`
  - `x == y` (**NOTE** the use of `==` and not `=`.)

Slide 6

Boolean Expressions, Continued

- *Boolean algebra* defines some operators on these values; the most important for us now are written in Scala as
  - `!` — "not", true if the operand is false.
  - `&&` — "and", true if both operands are true.
  - `||` — "or", true if either operand is true (or both are).
- Can use these to build up complex expressions. As with arithmetic expressions, use parentheses when in doubt. Examples:
  - `(x >= 0) && (x <= 10)` (What if we just write `0 <= x <= 10`?)
  - `!(x == y)` (though we could also just write `x != y`).

Slide 7

Boolean Expressions in Scala

- Scala has a type for boolean values (`Boolean`) with the obvious values.
- One thing to know is that the operators for "and" and "or" do not evaluate both operands if the value of the first operand determines the result.

Slide 8

Conditional Execution — `if/else`

- To execute a statement if an expression evaluates to true, use `if`:

```
if (x > 0)
    println("greater than zero")
```

- To execute one statement if an expression is true, another if it's false, use `if` and `else`:

```
if (x > 0)
    println("greater than zero")
else
    println("not greater than zero")
```

Slide 9

`if/else` in Expressions

- Similar rules apply within expressions, e.g.,

```
if (x < 0) -1 else 1
```

  has the value -1 if X is less than zero, 1 otherwise.

- Many programming languages have a similar construct but express it differently; in C and Java the equivalent expression is

```
(x < 0) ? -1 : 1
```

Slide 10

`if/else` and Blocks

- To execute a group ("block") of statements rather than just a single statement, use curly braces for grouping:

```
if (x > 0) {
    println("greater than zero")
    println("and that is good")
}
else {
    println("not greater than zero")
    println("and that is bad")
}
```

Slide 11

`if/else`, Continued

- What happens if you forget the braces? The program may still run, but it probably won't do what you meant.
- Several styles for where to put the curly braces. Which is best? Some people care; I say just pick one that's readable *and use it consistently*.

Slide 12

Example

- As a first example, revise the count-change program to
  - just print an error message if the number is not positive.
  - print only nonzero results.
- As another simple example, consider a program to calculate numeric and letter grades as described in the syllabus.

Slide 13

One More Conditional Construct — Match

- There are situations in which we have a lot of if/else code testing the same variable against various values. Somewhat repetitive, no? Some languages provide more-compact way to express this (e.g., `switch` in C and Java).
- Scala provides something more general — `match`. Allows matching specified variable with multiple conditions . . .

Slide 14

Match, Continued

- Simple example:

```
val c = readChar
c match {
  case 'a' => println("found a")
  case 'b' => println("found b")
  case _ => println("not a or b")
}
```

This is already more powerful than what some languages provide, in that you can match strings. Much more is possible. Details later. Discussed in chapter 6.

Slide 15

Minute Essay

- Make your best guess at writing a few lines of Scala code that ask the user for an integer, read it in, and print “positive”, “negative”, or “zero” based on the value read in.

Slide 16

Minute Essay Answer

- ```
  println("enter an integer")
  val x = readInt
  if (x > 0) println("positive")
  else if (x < 0) println("negative")
  else println("zero")
  ```
  ```