TextExcel Checkpoint 2

set and display cells
Due: 2015/02/25

In this checkpoint, you will implement the functionality to set and display values of different types for cells in your spreadsheet. To accomplish this, you will make use of subclasses that inherit from a `Cell` superclass.

Required features

- Users can set a cell's contents with the syntax **`<cell address> = <cell contents>`**.
- Users can view the current contents of a cell by typing the cell address by itself. It prints in the same **`<cell address> = <cell contents>`** format.
- When users `print` the entire spreadsheet (via checkpoint 1's `print` command), each cell's contents are shown in the printed sheet. Cell contents are centered.
- If cell contents are too long, they are trimmed to fit in 12 characters, with a > character to indicate that trimming has occurred (see example below)
- When users view a single cell, they see entire contents of the cell, untrimmed.
- Users may enter `Strings` (surrounded by quotes), `doubles`, or dates (MM/DD/YYYY)

Example session (user input in bold)

```
Welcome to TextExcel!
Enter a command: A1 = "Hello, world!"
Enter a command: B1 = 1/14/2015
Enter a command: C1 = 3.1415926
Enter a command: G6 = "shoo fly"
Enter a command: print

            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
------------+------------+------------+------------+------------+------------+------------+------------+
     1      |Hello, worl>| 01/14/2015 | 3.1415926  |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     2      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     3      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     4      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     5      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     6      |            |            |            |            |            |            |  shoo fly  |
------------+------------+------------+------------+------------+------------+------------+------------+
     7      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     8      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     9      |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
     10     |            |            |            |            |            |            |            |
------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: A1
A1 = "Hello, world!"
Enter a command: B1
B1 = 01/14/2015
Enter a command: exit
Farewell!
```

Required design

- Cell contents should be stored in subclasses of your `Cell` class for each type:
  - `StringCell`
  - `NumberCell`
  - `DateCell`
- `Cell` should have a method called `eval()` that takes a `Sheet` as a parameter and returns the (untrimmed) value to display in the spreadsheet. The `Sheet` parameter will be used later (Checkpoint 5) when we want to support evaluating cells containing formulas that refer to other cells (and therefore must look up their values in the sheet). `Cell` subclasses should each implement `eval()` in a way appropriate to their type.
- `Cell` should also have a `toString()` method that handles returning a value to display from the command loop. Note that for `NumberCell` and `DateCell`, this will have the same logic as the `eval()` method. However, a `StringCell` should have its value returned with quotation marks in `toString()`, but not in `eval()`, because the `StringCell` is printed with quotation marks at the command loop but not within the spreadsheet.

Required testing

Download Checkpoint2Test.java (if you're reading this on paper, go to the online version for the link) and add it to your project, then work to make the tests in it pass.

Concepts, reading references, and tips

Concepts used in this checkpoint include:

- Formatting output (4.3)
- Parsing (6.2, 6.3)
- Inheritance (superclasses and subclasses) (Chapter 9)

Tips:

- `String.substring()` (for trimming output when it's longer than 12 characters)
- `SimpleDateFormat` is a helper class that creates `Date` objects from `Strings`, as well as creates `Strings` from `Dates`. You should import `Date` from `java.util.Date`
  - `SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");`
  - `Date myDate = sdf.parse("02/03/2014");`
  - `String dateString = sdf.format(myDate);`
- `Double.parseDouble("12.34")` returns the double value `12.34`
- If a cell's length is exactly 12, don't trim it. Only trim if the cell has 13 or more characters.
- When you do trim, don't forget to trim the content to 11 characters, and append a “>” character