CS100J, Fall 2003, Assignment 1. Due on 16 September before midnight.

About submitting assignments

In this course, we will be using an electronic submission system developed by the CS department. As soon as possible --which means NOW-- please bring up this page in your browser. http://cms.csuglab.cornell.edu/ . You will probably asked for your Cornell id and password (using Kerberos). When you give them correctly and hit "enter", the Computer Science **C**ourse **M**anagement **S**ystem home page will appear in your browser. To the right, you should see the link "Courses CS100J FA03". If it is there, that means you are entered into the course list, and you shouldn't have problems submitting the first assignment later. **If the link is not there, that means that you are not in our course list!** You should immediately email ... <...@cornell.edu>, explaining that you are taking the course and want to be registered to submit assignments. Do this NOW --don't wait until it is time to submit the first assignment.

Purpose

Write several `JFrame` subclasses (customizations) in order to practice with writing classes, methods, method calls (including the use of arguments and parameters), and expressions. We will have taught all the material required for this assignment by the end of week 2.

Ground rules

You may work with one partner. If you do, get on the course management system several days before the assignment is due and follow directions for telling the system who your partner will be. Don't wait until the last minute to do this; do it several days before you want to submit the assignment.

This is (very, very roughly) a 10-hour assignment. Plan accordingly.

Plan to spend as much as an hour reading this handout so that you thoroughly understand what we are asking for. Do this **with your partner** before you start programming!

We strongly, strongly suggest that you and your partner alternate writing the classes (but with the other person watching and helping). You will both benefit from this, even if you find it feels like it is taking longer.

VITAL RULES

Here are some very specific rules that you **must** follow. If you break any of these, you will automatically get a zero on part or all this assignment.

1. You must use the capitalization that we specify. You can ensure this by copying identifiers from this handout, rather than typing them yourself.
2. You may not use the word `static` anywhere in assignment 1.

3. You may not use loops or if-statements.
4. Every method and class must be marked `public`, and every instance variable must be marked `private`.
5. Do not use `System.out.println` or `System.out.print` in any class that you write.

Subclasses of `JFrame`

You will write several subclasses of `JFrame`. Here are the sources of help that you can use, in no particular order: the course textbooks, your lecture notes, your programming partner, the course website, the Java APIs, the consultants in Carpenter, the TAs, and the instructor.

You should write a short javadoc comment at the top of each class that describes what the class is for. You should write a comment for each method. As a reminder, javadoc comments look like these:

```
/** this is a javadoc comment for class X */
public class X {

   /** this is a javadoc comment for method m.
       Put a blank line before me. */
   public void m() {
   }
}
```

The comment on a method MUST be a specification of the method. It should follow the guidelines given in Sec. 13.3.1 of the text. Also, use the specifications shown in Sec. 2.1 as examples. One way to get the specification is simply to copy it from the material below.

Here are the five subclasses that you should write:

- **DoublingHalvingFrame**: a subclass of `JFrame` that can double and halve its height. Has these additional methods:
  - `doubleHeight()`. Double this window's height.
  - `halveHeight()`. Cut this window's height in half.

  Hints:Write and check out one method at a time. In the Interactions pane, initialize a `JFrame` and figure out a statement that doubles the size of that window. Then write the subclass outline, copy the statement into the method, and change the variable name to "this". Also, check your lecture notes.
  Note: doubling and halving may stop working properly if the windows become too large for your screen or too small. Don't worry about this; your class need work only when the window is of a reasonable size.

- **SizingFrame**: a subclass of `JFrame` that can be resized in several different ways. Has these additional methods:
  - `changeSizeToLoc()`. Change this window's width to this window's X coordinate and set this window's height to this window's Y coordinate.
  - `addToSize(int dW, int dH)`. Add `dW` to this window's width and `dH` to this window's height.
  - `swapOrientation()`. Change this window's height to its width and its width to its height.

  Hints: write and test one method at a time. For example, after you write class `SizingFrame` and

method `changeSizeToLoc`, try this in the Interactions pane:

```
SizingFrame rw= new SizingFrame();
rw.show();
rw.setLocation(400, 400);
rw.changeSizeToLoc();
```

You should see the window change shape.

- **DateFrame**: a subclass of `JFrame` that can get its size and location from a `Date` object. Has these additional methods:
  - `setSizeAndLocToDate(Date d)`. Set this window's height to `120 + (d`'s year modulo `300)`, set this window's width to `120 + (d`'s month times `10)`, set this window's X coordinate to `50 + (d`'s hour times `5)`, set this window's Y coordinate to `50 + (d`'s seconds times `5)`, and set this window's title to the value of `"Date " + d.toString()`.

Hint: Look up "modulo arithmetic" in ProgramLive's glossary. Since the numbers you are working with are non-negative, you can compute `b modulo c` using `b % c` .

- **FramingFrame**: a subclass of `JFrame` that can create a partner frame. Has these additional methods:
  - `createPartner()`. Create a `JFrame` that is 50 pixels wider and 50 pixels taller than this frame and has its top-left corner 25 pixels to the left and 25 pixels up from this frame. The original frame should appear on top of this one, so that the partner sort of "frames" this original one.
  - `resetPartner()`. Resize the partner frame so that it is 50 pixels taller and wider than this frame and move it so that it "frames" this frame, as discussed in method createPartner. Again, this partner should appear behind this frame. This method is called only after `createPartner` has been called.
  - `getPartner()`. = The partner (null if not yet created). That is, return the name of the manilla folder of the partner frame.

Note: The first method creates a partner frame; the second one doesn't. It just moves and resizes the partner. For example, you, the user, can drag and resize the frame; then you can call resetPartner to fix its partner.
Hint: Save the partner frame as an instance variable. Draw a picture to work out the coordinates before you program it.
Hint: You have to show the partner frame in the two methods; otherwise, it won't become visible. You will also want to show the main frame itself, so that it appears before the partner frame.
Hint: Methods createPartner and resetPartner have much in common. See whether you can put most of the work in one of them and call that one from the other one.

- **CrossFrame**: a subclass of `JFrame` that can create a "cross" of frames around it. Has these additional methods:
  - `createCross()`. Create 4 `JFrame`s that are all half the size of this frame (i.e. the width is half the size and the length is half the size). One appears just above this frame and is centered horizontally. In the same way, one appears below and is centered horizontally, one appears to the left and is centered vertically, and one appears to the right and is centered vertically. Thus, the five frames look something like a cross.
  - `resetCross()`. Move and resize the 4 frames created in `createCross` so that they are all half the size of this frame and and are positioned as explained in the specification of method `createCross`. This method is called only after `createCross` has been called.
  - `getTop()`. = the partner that is above this frame (null if not yet created). That is, return the name of the partner frame above this one.

- `getBottom()`. = the partner that is below this frame (null if not yet created). That is, return the name of the partner frame below this one.
- `getLeft()`. = the partner that is to the left of this frame (null if not yet created). That is, return the name of the partner frame that is to the left of this one.
- `getRight()`. = the partner that is to the right of this frame (null if not yet created). That is, return the name of the partner frame that is to the right of this one.

Hint: save the partner frames as instance variables. Draw pictures to work out the coordinates.
Note: Suppose frame N is the one that goes above this frame. Suppose N's width is w and its height is h. Then the top-left corner of N MUST be h+1 pixels above and w/2 pixels to the right of the top-left corner of this frame. Similar statements hold for the other three frames.
Hint: Methods createCross and resetCross have much in common. See whether you can put most of the work in one of them and call that one from the other one.

Note: Windows apparently has a minimum width of 112 pixels. Even if you do `j.setSize(0, 0);`, the width is 112. You don't need to fix this, and you will not be penalized for this.

What to submit

Use link http://cms.csuglab.cornell.edu/ to get on the course website and submit the following five files.

- `DoublingHalvingFrame.java`,
- `SizingFrame.java`,
- `DateFrame.java`,
- `FramingFrame.java`,
- `CrossFrame.java`.
