CS 1331: Introduction to Object Oriented Programming

Homework 3 **Due:** January 31, 2014 @ 8:00 PM

Crazy Cipher

In cryptography, a cipher, is an algorithm that takes in some text, performs some transformations on the letters, and outputs an encoded message. We in CS 1331 have what we like to call the Crazy Cipher to send secret messages to each other without everyone knowing. Here's how it works:

- Our cipher only works on text with 5 letters
- The first letter stays the same
- The second letter is always shifted up 4 letters (shifting as in: a = e, f = k, etc...)
- The third letter is always shifted *back* 4 letters (e = a, k = f, etc..)
- The fourth and fifth letters are swapped

Your job is to encode a user's input and then decode another user input. The user will input the word to encode (it will always be of 5 letters) and decode (it might be different, but still 5 letters). Your program should look something like this:

```
Encode?
HaPpY
--> HeLYp
Decode?
HeLYp
--> HaPpY
```

Notes

- You are **NOT** allowed to use loops in this program. The point of this assignment is to get you familiar with the String API and manipulating Unicode values.
- Along the same lines as the above statement, you should have the String API open on your computer while you're doing this assignment. There are lots of helpful methods to use in there (substring, and charAt would be ones to look at first :))
- Your code only needs to encode and decode one word, and you can assume you will always get a 5 letter word and that each letter is different (as in the example above)
- In regards to shifting... Characters in Java are actually represented as Unicode numbers. When you have a 'char' variable, you actually have a number underneath the letter/symbol! So you can do things like:

  ```
  char mander = 'd';
  char meleon = (char)(mander + 1);
  ```

  Meleon would print to be 'e'. (If you are curious, the cast is because of a loss of precision error, can you think of why?)

- If you shift past the alphabet, you might get some sort of non-letter symbol, that's totally fine!

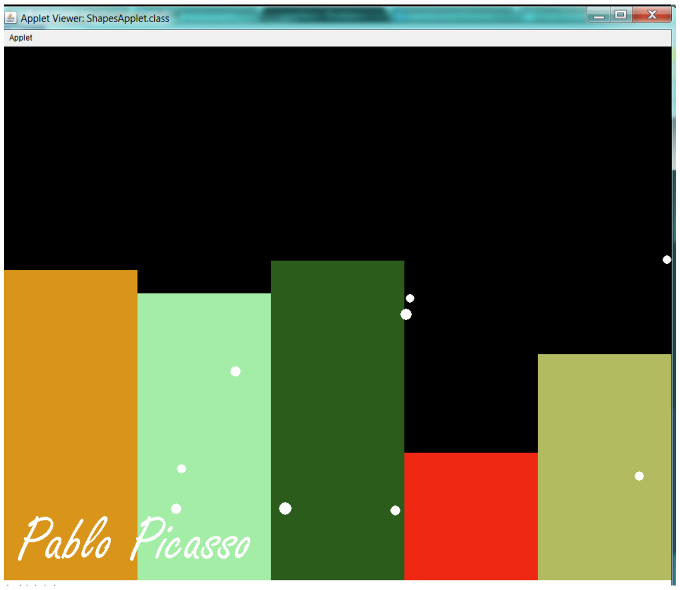
In Honor of Leon

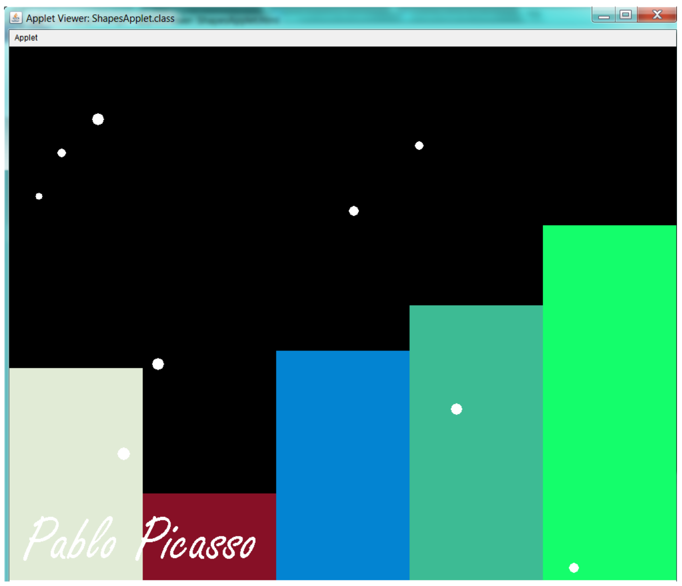
As you all know, class was cancelled for the majority of last week due to Winter Storm Leon. For over 48 hours, your phone was likely flooded with snapchats of the snow, texts about the snow, and you may have recieved a phone call or two about the snow. You got at least 20 emails from Georgia Tech about to the snow. Your Facebook news feed was completely filled with statuses about the snow (or the traffic). It's safe to say that, for those few days, everything was SNOW. To preserve these treasured memories, you will now create an applet that will show a picture of the Atlanta skyline in all of its snowy wonder.

Your job is to create an applet that randomly generates a city skyline. Your city must have

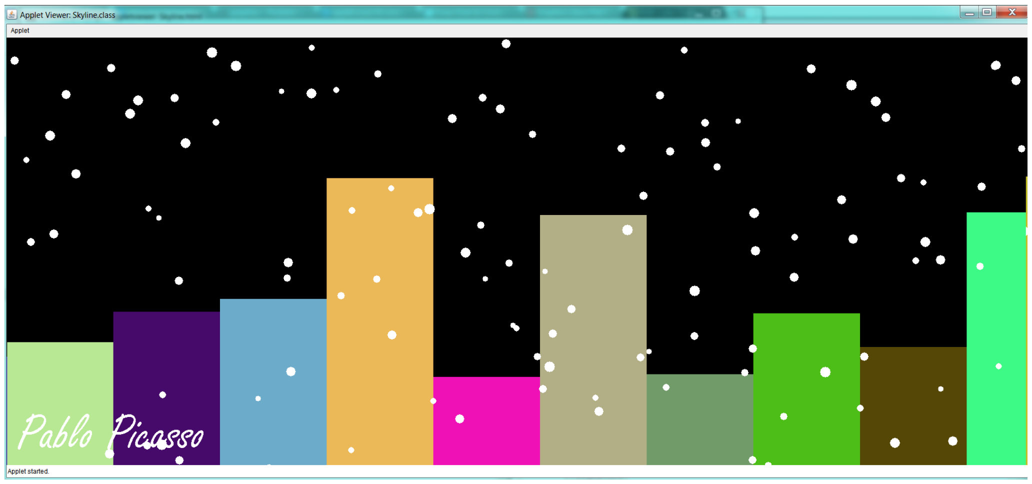
- A black sky with at least 10 snowflakes of random sizes, positioned randomly.
- At least 5 buildings (rectangles) with randomly generated colors and heights. This MUST span the entire width of your applet.
- Your name, in white, in the bottom-left corner.

Your pictures should look something like this.





You are encouraged to make this as beautiful and snowy as your heart desires.



- The heights of your buildings should range from 100 to 600 pixels.
- The snowflakes should be circles, ranging anywhere from 10 to 20 pixels in diameter.
- The height of your applet must be 800 pixels, whereas the width can be anywhere between 1000 and 2000 pixels.
- The maximum red, green, and blue values in RGB coloring is 255.
- If you are having problems with AppletViewer, please try using jGrasp to display your applet!

Your buildings are allowed to be different widths and/or overlap, as long as none of the background shows through.

Checkstyle

You have been provided with a Checkstyle Jar file and the CS 1331 Checkstyle configuration file (`cs1331-checkstyle.xml`). We do this to make sure you are following the proper style guidelines as posted in the Style Guide under T-Square resources. Please refer to these guidelines for any help as you are writing your programs.

We will run checkstyle on all the Java source files you submit. You can run Checkstyle with the command below (in the directory containing all your Java source files):

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by using the command below and subtracting 2 from the number printed (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
2
```

Also means no errors.

If you have any questions about a Checkstyle error, please let a TA know! They should be pretty self-explanatory and you should be able to fix them easily. We will be taking points off for checkstyle on this assignment, so please make sure you run it!

Turn-in Procedure

Submit all of the Java source files you created to T-Square and your HTML page for your Applet. Do not submit any compiled bytecode (`.class` files). When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code with Checkstyle!

Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1]Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is 8PM Friday. Do not wait until the last minute!