Live Coding

Oct 16.2015

Please pair with someone & collaborate. Please share unit-test cases to make sure that the code is solid. Each section lists points earned(100 means superb). We're going to making "Ascii/Unicode" animations today.

A. Grayscale2Unicode – Code 10 pts, Unit Tests 15 pts

a. Create a function that takes in a positive number N between `0-84` inclusive and returns a single unicode character in a string according to the following table: Goto: https://github.com/JonathanRitchey03/GrayscaleUnicodeArt and use the string in the readme file there.
Make 3 unit tests to verify it works properly. See appendix at end on how to add unit tests.

0 → \u2588 ☐
1 → \u2589 ☐
2 → \u258A ☐
3 → \u258B ☐
4 → \u2593 ☐
5 → \u25A9 ☐
6 → \u25A6 ☐
7 → \u25A4 ☐
8 → \u25A7 ☐
9 → \u25A8 ☐
10 → \u25C9 ◉
11 → \u25A3 ▣
12 → \u25C8 ◈
13 → \u2592 ☐
14 → \u2591 ☐
15 → $
16 → @
17 → B
18 → %
19 → 8
20 → &
21 → W
22 → M
23 → #
24 → *
25 → o
26 → a
27 → h
28 → k
29 → b
30 → d
31 → p
32 → q
33 → w
34 → m
35 → Z
36 → O
37 → 0
38 → Q
39 → L
40 → C
41 → J
42 → U
43 → Y
44 → X
45 → z
46 → c
47 → v
48 → u
49 → n
50 → x
51 → r
52 → j
53 → f
54 → t
55 → /
56 → \
57 → |
58 → (
59 →)
60 → 1
61 → {
62 → }
63 → [
64 →]
65 → ?
66 → -
67 → _
68 → +
69 → ~
70 → <
71 → >
72 → I
73 → !
74 → l
75 → I
76 → ;
77 → :
78 → ,
79 → "
80 → ^
81 → `
82 → '
83 → .
84 →

B. GrayscaleUnicodeCanvas – Code 20 pts, Unit Tests 35 pts

a. Create a class that will represent a 2D array of grayscale values from 0..255(uint8) and render to unicode values. Write the render function using grayScale2Unicode(uint8 gsvalue) method made in problem A.

Note: The backing array is one-dimensional. That is intentional. Math is done to map from 2D to 1D. Please let me know if there are any questions about how that works. Here's the basic skeleton:

```
public class GSCanvas {
      short mArray[];
      int mWidth,mHeight;
      // constructor
      public GSCanvas(int width,int height) {
            // assert 0<width<MAX_WIDTH,0<height<MAX_HEIGHT,
            // else default to width=height=DEFAULT_DIM
            mArray = new short[width * height];
            mWidth = width;
            mHeight = height;
      }
      public void set(int x, int y, short grayscaleValue) {
            // assert 0<x<mWidth, 0<y<mHeight, else do nothing
            mArray[y*width + x] = grayscaleValue;
      }
      public void fillRect(int x0,int y0,int w,int y,short value) {
            for(int y = y0; y < y0+h; y++)
                  for(int x = x0; x < x0+w; x++)
                        set(x,y,value);
      }
      public short get(int x, int y) {
            // assert 0<x<mWidth, 0<y<mHeight, else do nothing
            return mArray[y*width + x];
      }
      public String render() {
            // map 0..255 to 0..84 and use grayScale2Unicode function
            // write code for this
      }
}
```

Make unit tests to verify each method works properly. Please see me for help if there's difficulties.

C. AverageMethod – Code 20 pts, Unit Test 25 pts

a. Add an average method to the class, that creates a new empty array. Now for each cell of the new array, take the sum of all the cells around it. Specifically for a new cell at x,y:

```
int sum = get(x-1,y-1) + get(x,y-1) + get(x+1,y-1) +
          get(x-1,y) + get(x+1,y) +
          get(x-1,y+1) + get(x,y+1) + get(x+1,y+1);
```

Now let the new cell's value equal the average of all the neighbors above.
After that, copy the newArray over mArray. So now mArray has the averaged array.

```
public class GSCanvas {
      …
      public void average() {
        short newArray[] = new short[mWidth * mHeight];
        for ( int y = 0; y < mHeight; y++ ) {
            for ( int x = 0; x < mWidth; x++ ) {
                int sum =    get(x-1,y-1) + get(x,y-1) + get(x+1,y-1) +
                             get(x-1,y) + get(x+1,y) +
                             get(x-1,y+1) + get(x,y+1) + get(x+1,y+1);
                  …

            // please complete
    }
```

D. Beautiful Animation – Code 30 pts

a. Now in the main class add code that randomly draws a fill rect with a random value into the array then averages it using the average method. Render to the screen. Put this in a loop, clear the screen, then pause for 1/10th of a second. Keep repeating. Show me your animation when finished. Try different ideas...
Here's my implementation for reference...

```
import java.util.Random;
public class Main {
    public static void main(String[] args) {
       GSCanvas canvas = new GSCanvas(128,28);
```

```
        Random random = new Random();
        for ( int k = 0; k < 10; k++ ) {
            for (int i = 0; i < 20; i++) {
                if ( random.nextInt(2) == 0 ) {
                    canvas.fillRect(random.nextInt(127),
                                    random.nextInt(40),
                                    random.nextInt(30),
                                    random.nextInt(30),
                                    (short) random.nextInt(127));
                }
                for (int j = 0; j < 2; j++) {
                    canvas.average();
                }
                System.out.print(canvas.render());
                //pressAnyKeyToContinue();
                try { Thread.sleep(200); } catch(InterruptedException e) {}
                clearScr();
            }
        }
    }
    private static void pressAnyKeyToContinue()
    {
        System.out.println("Press any key to continue...");
        try {
            System.in.read();
        } catch(Exception e)
        {}
    }
    public static void clearScr() {
        final String ANSI_CLS = "\u001b[2J";
        final String ANSI_HOME = "\u001b[H";
        System.out.print(ANSI_CLS + ANSI_HOME);
        System.out.flush();
    }
}
```

E. If you get really stuck I've placed my IntelliJ project for reference up at:

a. Find the github repo at:
https://github.com/JonathanRitchey03/GrayscaleUnicodeArt

b. The source code for Main and GSCanvas are at:
https://github.com/JonathanRitchey03/GrayscaleUnicodeArt/tree/master/src/com/company

Please let me know if I can help...

F. Bonus challenge

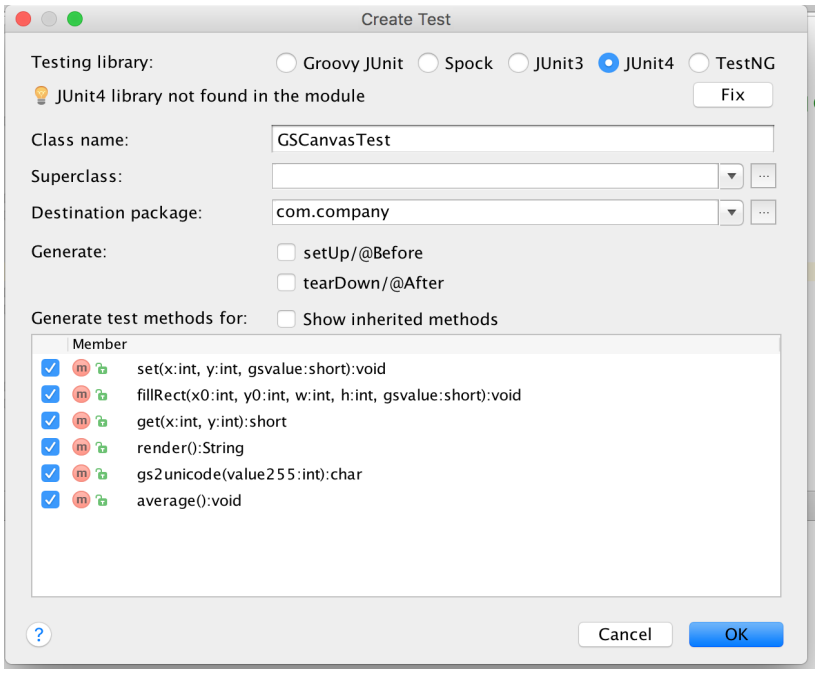
If you'd like an additional challenge. Try putting random values into the array and animate a quicksort on those values using the same visualization technique as in challenge E! Wikipedia has an excellent page with the pseudocode for Quicksort... It'll probably look really interesting...
https://en.wikipedia.org/wiki/Quicksort

```
quicksort(A, lo, hi)
  if lo < hi
    p = partition(A, lo, hi)
    quicksort(A, lo, p - 1)
    quicksort(A, p + 1, hi)

partition(A, lo, hi)
    pivot = A[hi]
    i = lo //place for swapping
    for j = lo to hi - 1
        if A[j] <= pivot
            swap A[i] with A[j]
            i = i + 1
    swap A[i] with A[hi]
    return i
```

Appendix – How to add Unit Tests to IntelliJ

1. In IntelliJ select the file you want to add tests to and press Command-Shift-T. (⌘ + shift + T)
2. Select "Create New Test".
3. On the pop that comes up, select JUnit 4. Now click on the "Fix" button so it includes the JUnit 4 package.
4. Click on all the methods. (see screenshot)



Here are some example unit tests:

```
@Test
public void testingCrunchifyAddition() {
assertEquals("Here is test for Addition Result: ", 30, addition(27, 3));
}

@Test
public void testingHelloWorld() {
assertEquals("Here is test for Hello World String: ", "Hello + World", helloWorld());
}
```

Appendix – How to add Unit Tests in PyCharm

https://confluence.jetbrains.com/display/PYH/Creating+and+running+a+Python+unit+test

1. In IntelliJ select the file you want to add tests to and press Command-Shift-T. (⌘ + shift + T)
2. Click on the check mark on all the method then press "OK". See screenshot:

Here are some example test cases:

```
def test_get_team_and_score_from_string(self):

    self.assertEqual(rank_teams.get_team_and_score_from_string("My Team 5"), ("My Team", '5'))
    self.assertEqual(rank_teams.get_team_and_score_from_string("My Team 5        "), ("My Team", '5'))

def test_update_rank_dict_for_team_by_points(self):

    rank_dict = {"teamA": 1}
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamA", 1)
    self.assertEquals(rank_dict, {"teamA": 2})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamA", 0)
    self.assertEquals(rank_dict, {"teamA": 2})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamB", 0)
    self.assertEquals(rank_dict, {"teamA": 2, "teamB": 0})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamB", 3)
    self.assertEquals(rank_dict, {"teamA": 2, "teamB": 3})
```

Appendix – How to add Unit Tests in Visual C#

Check online. One link I found was:

http://www.codeproject.com/Articles/391465/Creating-Unit-tests-for-your-csharp-code

Appendix – How to add Unit Tests in JavaScript

If using WebStorm, JetBrains has support for this.

http://www.codeproject.com/Articles/391465/Creating-Unit-tests-for-your-csharp-code