Building WordCloud Program Step-by-Step

In your Processing folder, create a subfolder called WCInc.

WCInc01
Open Processing and immediately save the sketch to the WCInc folder as WCInc01. Create two global PFont variables: fontImpact and fontMistral. Use createFont() to initialize these variables with the Windows system fonts Impact and Mistral. Set the size of the window to 600W x 400H.

Write a method called bgTrans(), in which you draw the window’s background using a transparent red. Use a global final color variable called RED_T to do this. The transparency value should be 12. Call bgTrans() in draw().

WCInc02
1. Create a class called PWord. Write a constructor with no body and no parameters. Write 2 class methods: write1() and write2().

2. write1() will use the Impact font. It will display the word “Again” so that its baseline is at the middle of the window’s left edge. The size of the font should be 48, the color should be white, and the word should display LEFT aligned.

3. write2() will use the Mistral font. It will display the word “Zipline” in the middle of the window’s bottom edge. The size of the font should be 64, the color should be yellow, and the word should display CENTER aligned.

To figure out how to specify the exact value for the vertical position so that the descender of the “p” is visible, look up the method textDescent().

4. Create two global PWord variables: word1 and word2. Initialize each with a new PWord object in setup(). Use word1 and word2 in draw() to display both words.

WCInc03
Create 7 class variables.

A PFont variable called font.
A String variable called word.
An int variable called size.
Two float variables called x and y.
A color variable called clr.
An int variable called align.

Write a method called write() – modeled after write1() and write2() – that uses these 7 class variables instead of the hard-coded values that those two methods use.

Create a third global PWord variable: word3. Initialize it with a new PWord object. Then – also in setup() – use the word3 variable with dot syntax to initialize each of its 7 class variables with values that will display the word “Elephant” in blue so that its baseline is at the middle of the right border, RIGHT aligned, using the font Rage Italic, text size 80.
Building WordCloud Program Step-by-Step

WCInc04
In setup(), declare/create 7 local variables corresponding to the 7 class variables in PWord. They will have the same names as the class variables, with the addition of an ending: “Value”, e.g. fontValue, wordValue, etc.

Initialize these 7 variables with the values you had assigned directly to the class variables in WCInc03, e.g. sizeValue = 80. Then substitute these local variables for the hard-coded values. Below is the complete example for alignValue:

```plaintext
// keep all of the local variable declaration/initialization statements together
int alignValue = RIGHT;

// keep all of the class variable initialization statements together
word3.align = alignValue;
```

There will be no difference from WCInc03 when you run the program.

WCInc05
In PWord, make a duplicate copy of the constructor, but comment it out (you will use it later on).

From setup(), cut/delete the 14 lines of code that you made changes to in WCInc04, and paste it into the PWord constructor. Replace the 7 instances of word3 with this.

There will be no difference from WCInc03/04 when you run the program.

WCInc06
In PWord, promote all 7 local variables to parameters, ONE AT A TIME as follows:

1. Change each local variable name by replacing its ending “Value” with “PassIn”, e.g. fontValue becomes fontPassIn.

2. Move the left side of the declaration/initialization in between the parentheses. Then move the right side (value) of the declaration/initialization to the constructor in setup(). Below is how your final code should look for the font parameter:

```plaintext
// in PWord.pde
PWord(PFont fontPassIn) {
    this.font = fontPassIn;
}

// in WCInc06.pde
word3 = new PWord(fontRage);
```

3. Comment the second constructor back in (i.e. remove the comment-out slashes).

When done, you will have TWO constructors, one with no parameters and one with 7 parameters.

There will be no difference from WCInc03/04/05 when you run the program.
Building WordCloud Program Step-by-Step

WCInc07
1. Add the line below to write2() as its last line:

   println( textDescent() );

Run the program and note the value of textDescent().

2. In setup(), initialize word1 and word2 with the PWord constructor that uses 7 parameters. Both words will display exactly as before (using the same color, size, font, position, etc.) Note that textDescent() no longer returns the correct value. Replace it with the number value you found above in (1).

3. Delete the 2 methods write1() and write2(). The program now contains two bugs in draw(). Fix these so both words display.

   There will be no difference from WCInc03/04/05/06 when you run the program.

WCInc08
In PWord, modify write() so that it can draw a word at any angle. In the main module, draw word2 at a 90 degree angle, align = RIGHT. The textDescent() value is no longer needed for “Zipline”: Remove it!

WCInc09
1. In PWord, write a (void) method named placeRandomlyOnEdge(). In the body of this new method, write an if-else statement with FOUR (4) BRANCHES. The condition for each branch (what’s inside the parentheses) should figure out on which edge the word is positioned. It should then adjust one of the coordinates so that the word will appear randomly anywhere on that same edge each time the program starts up.

2. Call/Use this method at the bottom of the body of the constructor (i.e. as its very last statement).

   Below are 3 examples of how the window might appear when the program starts up:

WCInc10
In placeRandomlyOnEdge(), notice that the bodies of the 4 branches of the if-else statement contain duplicate statements. 2 branches contain identical statements; the other 2 branches do too.

Modify placeRandomlyOnEdge() so that its if-else statement has only TWO BRANCHES. You do this for each pair of duplicate branches by (a) eliminating one of the duplicate branches, but (b) combining both conditions into a COMPOUND boolean expression (one that uses AND ( && ) or OR ( || )). Figure out whether you should use && or ||.

   There will be no difference from WCInc09 when you run the program.
Building WordCloud Program Step-by-Step

WCInc11
In PWord, modify write() so that each word will spin clockwise at the rate of one complete revolution EVERY 12 SECONDS. Remember that the default frameRate is 60 frames/second and that there are 360° in a circle.

WCInc12
1. In PWord, create two more class variables. They will be called xOpp and yOpp and they will be of type float, like x and y.

2. Write a method called calcOppositeEdge() that will calculate the corresponding point on the opposite edge, For example:

   If a word were on the TOP edge, xOpp and yOpp would be the coordinates of the point on the BOTTOM edge directly below it, so that if you were to draw a line between the two points, it would be vertical.

   If a word were on the LEFT edge, xOpp and yOpp would be the coordinates of the point on the RIGHT edge directly across from it, so that if you were to draw a line between the two points, it would be horizontal.

3. Call/Use this method at the bottom of the body of the constructor (i.e. as its very last statement).

4. Modify write() so that it uses xOpp and yOpp instead of x and y.

WCInc13
Modify write() as follows:

Each word will travel – still spinning – from its starting position at (x,y), either horizontally or vertically, across the window to the opposite edge at (xOpp, yOpp), and then beyond without stopping. The travel time to reach the opposite edge should be 3 SECONDS.

Use TWO time variables (of type float): timeReal and time. timeReal will be calculated directly from frameCount and the frame rate (60.0). time is the variable that will be used in the expressions for calculating a word’s position. Depending upon what you will be doing in this and the subsequent programs, time may or may not have the same value as timeReal.

WCInc14
Modify write() as follows:

No change to each word as it travels across the window. However, once each word reaches the opposite edge, it will stop and remain there, continuing to spin in place.
Building WordCloud Program Step-by-Step

**WCInc15**
Modify `write()` as follows:

As each word travels across the window, it will stop at the halfway point between the edges for 3 SECONDS. It will then continue to the opposite edge.

Once a word reaches the opposite edge, the animation will repeat. That is, the word will immediately appear at the starting edge and travel across the window again (and again), stopping at the halfway point.

**WCInc16**
Modify `write()` as follows:

Same behavior as **WCInc15**, except that once the word reaches the opposite edge, it will bounce and move in the reverse direction. When a word reaches the halfway point between the edges, regardless of which direction it is traveling, it will stop for 3 seconds.