Word Clouds
Inheritance

- **Superclass (base class)** – higher in the hierarchy
- **Subclass (child class)** – lower in the hierarchy
- A subclass is derived from from a superclass
- Subclasses **inherit** the **fields** and **methods** of their superclass.
  - I.e. subclasses automatically "get" stuff in superclasses
- Subclasses can **override** a superclass method by redefining it.
  - They can replace anything by redefining locally
// Ellipse base class
class Ellipse {
    float X;
    float Y;
    float W;
    float H;

    // Ellipses are always red
    color fillColor = color(255, 0, 0);

    Ellipse(float X, float Y, float W, float H) {
        this.X = X;
        this.Y = Y;
        this.W = W;
        this.H = H;
    }

    void draw() {
        ellipseMode(CENTER);
        fill(fillColor);
        ellipse(X, Y, W, H);
    }
}

// Circle derived class
class Circle extends Ellipse {
    Circle(float X, float Y, float D) {
        super(X, Y, D, D);

        // Circles are always green
        fillColor = color(0, 255, 0);
    }

    // The extends keyword creates
    // hierarchical relationship between
    // classes.

    // The Circle class gets all fields and
    // methods of the Ellipse class,
    // automatically.

    // The super keyword refers to the
    // base class in the relationship.

    // The this keyword refers to the
    // object itself.
}

Graphics.pde
// Graphics
Ellipse e = new Ellipse(150, 250, 150, 50);
Circle c = new Circle(350, 250, 75);

void setup() {
    size(500, 500);
    smooth();
}

void draw() {
    e.draw();
    c.draw();
}
// Graphics2
Ellipse[] e = new Ellipse[20];

void setup() {
    size(500, 500);
    smooth();

    for (int i=0; i<e.length; i++) {
        float X = random(0, width);
        float Y = random(0, height);
        float W = random(10, 100);
        float H = random(10, 100);

        // Ellipses are Circles are
        // stored in the same array
        if (random(1.0) < 0.5)
            e[i] = new Ellipse(X,Y,W,H);
        else
            e[i] = new Circle(X,Y,W);
    }
}

void draw() {
    for (int i=0; i<e.length; i++)
        e[i].draw();
}

Ellipses and Circles in the same array!
// Ellipse base class
class Ellipse {

    float X;
    float Y;
    float W;
    float H;

    // Ellipses are always red
color fillColor =
        color(255,0,0);

    Ellipse(float X, float Y, float W, float H)
    {
        this.X = X;
        this.Y = Y;
        this.W = W;
        this.H = H;
    }

    void draw() {
        ellipseMode(CENTER);
        fill(fillColor);
        ellipse(X, Y, W, H);
    }

    // Do nothing
    void mousePressed() {}
}

// Circle derived class
class Circle extends Ellipse {

    Circle(float X, float Y, float D) {
        super(X, Y, D, D);

        // Circles are always green
        fillColor = color(0,255,0);
    }

    // Change color of circle when clicked
    void mousePressed() {
        if (dist(mouseX, mouseY, X, Y) < 0.5*W)
            fillColor = color(0,0,255);
    }
}

• The mousePressed behavior of the Circle class overrides the default behavior of the Ellipse class.
// Graphics3
Ellipse[] e = new Ellipse[20];

void setup() {
  size(500, 500);
  smooth();

  // Stuff removed …
}

void draw() {
  for (int i=0; i<e.length; i++)
    e[i].draw();
}

void mousePressed() {
  for (int i=0; i<e.length; i++)
    e[i].mousePressed();
}
What is a word cloud?

Source:
http://www.huffingtonpost.com/2013/09/01/1100-words-to-describe-your-summer00-words-to-describe-you_n_3853071.html
Text Processing

How to go from this…

Annie Schugart @AnnieSchugart - 31 Aug 2013
@HuffPostTeen Living once-in-a-lifetime experiences! Does that hyphenated word count? #mySummerIn3Words

...to this?

Aly @LeRoyAlly - 31 Aug 2013
@HuffPostTeen 1D, YouTube, Twitter #mySummerIn3Words

Paige McKenzie @hauntedsunshine - 31 Aug 2013
Worked very hard RT @HuffPostTeen: If you had to describe your summer using only 3 words, what would they be?? #mySummerIn3Words

Aysha @ayshasworld - 31 Aug 2013
@HuffPostTeen @HuffPostTeen sleep, school and eating #mySummerIn3Words

Isabel Song @IsabelSong - 31 Aug 2013
Study, sleep, Chicago. :) #mySummerIn3Words @HuffPostTeen
Text Processing

- Acquire - Obtain the data from some source
- Parse - Give the data some structure, clean up
- Filter - Remove all but the data of interest
- Mine - Use the data to derive interesting properties
- Represent - Chose a visual representation
- Refine – Improve to make it more visually engaging
- Interact - Make it interactive

### Data Visualization Process

### Text Visualization

- Source = Document
- Parse = Words
- Filter = Word Set with counts
- Mine = Get relevant words
- Represent = Fonts/Placement
- Refine/Interact
What's a string?

Characters enclosed by double quotes

"this is a String"
"   this String starts with spaces"
"12345"
"the above String is made up of digit characters"

Print Strings to the Console using println()

println( "The mouse was pressed" );
Strings are Objects

Defined using a class
Have fields, methods, one or more constructors

String objects hold an array of 'chars'
What's a char?
- A character enclosed by single quotes ('A')

String msg = "I Love CS 110!";
Making Strings

- Declaring String objects with no chars
  ```java
  String myName;
  String myName = new String();
  ```

- Declaring String objects init'd w/ char array
  ```java
  String myName = "Dianna";
  String myName = new String("Dianna");
  ```
Chars are encoded by bytes

**ASCII**
- *American Standard Code for Information Interchange*
- An early character encoding standard
- glyph <-> byte mapping
- 127 characters
- Forms the basis of new encoding standards
- **Unicode**: more than 109,000 characters covering 93 scripts

**Note:**
- Numbers are different than the digit characters
- Includes special characters and punctuation
<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul)</td>
<td>0</td>
<td>(dc4)</td>
<td>20</td>
<td>(</td>
<td>40</td>
<td>&lt;</td>
<td>60</td>
<td>P</td>
<td>80</td>
<td>d</td>
<td>100</td>
<td>x</td>
<td>120</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>(nak)</td>
<td>21</td>
<td>)</td>
<td>41</td>
<td>=</td>
<td>61</td>
<td>Q</td>
<td>81</td>
<td>e</td>
<td>101</td>
<td>y</td>
<td>121</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>(syn)</td>
<td>22</td>
<td>*</td>
<td>42</td>
<td>&gt;</td>
<td>62</td>
<td>R</td>
<td>82</td>
<td>f</td>
<td>102</td>
<td>z</td>
<td>122</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>(etb)</td>
<td>23</td>
<td>+</td>
<td>43</td>
<td>?</td>
<td>63</td>
<td>S</td>
<td>83</td>
<td>g</td>
<td>103</td>
<td>{</td>
<td>123</td>
</tr>
<tr>
<td>(eot)</td>
<td>4</td>
<td>(can)</td>
<td>24</td>
<td>,</td>
<td>44</td>
<td>@</td>
<td>64</td>
<td>T</td>
<td>84</td>
<td>h</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(enq)</td>
<td>5</td>
<td>(em)</td>
<td>25</td>
<td>-</td>
<td>45</td>
<td>A</td>
<td>65</td>
<td>U</td>
<td>85</td>
<td>i</td>
<td>105</td>
<td>}</td>
<td>125</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>(sub)</td>
<td>26</td>
<td>.</td>
<td>46</td>
<td>B</td>
<td>66</td>
<td>V</td>
<td>86</td>
<td>j</td>
<td>106</td>
<td>~</td>
<td>126</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>(esc)</td>
<td>27</td>
<td>/</td>
<td>47</td>
<td>C</td>
<td>67</td>
<td>W</td>
<td>87</td>
<td>k</td>
<td>107</td>
<td>(del)</td>
<td>127</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>(fs)</td>
<td>28</td>
<td>0</td>
<td>48</td>
<td>D</td>
<td>68</td>
<td>X</td>
<td>88</td>
<td>l</td>
<td>108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>(gs)</td>
<td>29</td>
<td>1</td>
<td>49</td>
<td>E</td>
<td>69</td>
<td>Y</td>
<td>89</td>
<td>m</td>
<td>109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>(rs)</td>
<td>30</td>
<td>2</td>
<td>50</td>
<td>F</td>
<td>70</td>
<td>Z</td>
<td>90</td>
<td>n</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>(us)</td>
<td>31</td>
<td>3</td>
<td>51</td>
<td>G</td>
<td>71</td>
<td>[</td>
<td>91</td>
<td>o</td>
<td>111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>(sp)</td>
<td>32</td>
<td>4</td>
<td>52</td>
<td>H</td>
<td>72</td>
<td>\</td>
<td>92</td>
<td>p</td>
<td>112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>!</td>
<td>33</td>
<td>5</td>
<td>53</td>
<td>I</td>
<td>73</td>
<td>]</td>
<td>93</td>
<td>q</td>
<td>113</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>&quot;</td>
<td>34</td>
<td>6</td>
<td>54</td>
<td>J</td>
<td>74</td>
<td>^</td>
<td>94</td>
<td>r</td>
<td>114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>#</td>
<td>35</td>
<td>7</td>
<td>55</td>
<td>K</td>
<td>75</td>
<td>_</td>
<td>95</td>
<td>s</td>
<td>115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>$</td>
<td>36</td>
<td>8</td>
<td>56</td>
<td>L</td>
<td>76</td>
<td>`</td>
<td>96</td>
<td>t</td>
<td>116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>%</td>
<td>37</td>
<td>9</td>
<td>57</td>
<td>M</td>
<td>77</td>
<td>a</td>
<td>97</td>
<td>u</td>
<td>117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>&amp;</td>
<td>38</td>
<td>:</td>
<td>58</td>
<td>N</td>
<td>78</td>
<td>b</td>
<td>98</td>
<td>v</td>
<td>118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>'</td>
<td>39</td>
<td>;</td>
<td>59</td>
<td>O</td>
<td>79</td>
<td>c</td>
<td>99</td>
<td>w</td>
<td>119</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**String class methods**

- `charAt(index)`
  - Returns the character at the specified index

- `equals(anotherString)`
  - Compares a string to a specified object

- `equalsIgnoreCase(anotherString)`
  - S/A ignoring case (i.e. 'A' == 'a')

- `indexOf(char)`
  - Returns the index value of the first occurrence of a character within the input string

- `length()`
  - Returns the number of characters in the input string

- `substring(startIndex, endIndex)`
  - Returns a new string that is part of the input string

- `toLowerCase()`
  - Converts all the characters to lower case

- `toUpperCase()`
  - Converts all the characters to upper case

- `concat(anotherString)`
  - Concatenates String with anotherString
String s1 = "abcdefg";
println( s1.charAt(0) );

String s1 = "abcdefg";
String s2 = "abcdefg";
if (s1.equals(s2)) println("They are equal");

String s1 = "abcdefg";
println( s1.indexOf('c') );

String s1 = "abcdefg";
println( s1.substring(2, 5) );

println( "abcdefg".length() );

println( "abcdefg".toUpperCase() );
Comparing Strings: Always use `equals()`

- Never use `==` ... Why?
  - String are objects
  - The `==` operator checks that two items are identical
  - Two objects can contain the same data, but be different object instances
  - The `==` operator tests that the two objects are the same object ... generally, that's not what we want
  - The `equals()` method tests the data of the two String objects for equality
## Other forms of `indexOf()`

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| int         | `indexOf(int ch)`  
|             | Returns the index within this string of the first occurrence of the specified character. |
| int         | `indexOf(int ch, int fromIndex)`  
|             | Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index. |
| int         | `indexOf(String str)`  
|             | Returns the index within this string of the first occurrence of the specified substring. |
| int         | `indexOf(String str, int fromIndex)`  
|             | Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |
### Other forms of `substring()`

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| String  | `substring(int beginIndex)`  
|         | Returns a new string that is a substring of this string. |
| String  | `substring(int beginIndex, int endIndex)`  
|         | Returns a new string that is a substring of this string. |
Digit chars in a String are not integers

String s = "12345";

void setup() {
    char myChar = s.charAt(1);
    byte myByte = byte(myChar);
    print(myChar);
    print(": ");
    println(myByte);
}

Result:
2: 50
void setup() {
    String s1 = "Hello";
    String s2 = "World";
    String s3 = one + " " + two;
    println(s3);
}

void setup() {
    String s1 = "She is number ";
    String s2 = " in computer science.";
    String s3 = s1 + 1 + s2;
    println(s3);
}

Numbers are converted to Strings prior to concatenation
Special chars in a String using escape char(\)
Use the escape character to embed special characters in a String

'\n' new line
'\t' tab

void setup() {
    println("This is line 1\nThis is line 2");
}
Strings can be held by Arrays
- (Just like any other object or primitive type)

```java
String[] tokens = new String[5];

void setup() {
    tokens[0] = "one";
    tokens[1] = "two";
    tokens[2] = "three";
    tokens[3] = "four";
    tokens[4] = "five";

    println(tokens);
}
```
Strings can be held by Arrays

- Initialized when declared

```java
String[] tokens = new String[] {"one", "two", "three", "four", "five"};

void setup() {
    println(tokens);
}
```
Strings can be held by Arrays
- Not initialized

```java
String[] tokens = new String[5];

void setup() {
  println(tokens);
}
```
Built-in String functions (not methods)

- `split( bigString, splitChar)`
  - Breaks a String into a String Array, splitting on splitChar
  - Returns new String Array

- `splitTokens( bigString, splitCharString )`
  - Breaks a String into a String Array, splitting on any char in splitCharString

- `join( stringArray, joinChar )`
  - Builds a new String by concatenating all Strings in stringArray, placing joinChar between each
  - Inverse of split() function

- `nf( intValue, digits )`
- `nf( floatValue, left, right )`
  - Formats a number as a String

- `trim( theString )`
  - Removes whitespace from the beginning and end of theString

- `text( theString, x, y )`
- `text( theString, x, y, width, height )`
  - Draws theString on the sketch at (x, y)
Split a String based on a single or multiple separator chars

String s1 = "12, 34, 56";
String[] as;

void setup() {
    as = split(s1, ",");
    //as = trim(as);
    println(as);
}

String s1 = "Data: 12, 34, 56";
String[] as;

void setup() {
    as = splitTokens(s1, ":,\"\"); //as = trim(as);
    println(as);
}
Join a String Array with a join char

String[] as = new String[] {"one", "two", "buckle my shoe"};

void setup() {
    String s1 = join( as, " | " );
    println( s1 );
}

one | two | buckle my shoe
Numbers can be formatted as Strings

String s1 = "She is the";
String s2 = "programmer.";

phrase = s1 + nf(7, 3) + " " + s2;
// nf( integer, number of digits )
// "She is the 007 programmer."

phrase = s1 + nf(3.14159,3, 2) + " " + s2;
// nf( float, digits before decimal, digits after decimal )
// "She is the 003.14 programmer."
// Sketch 7–1: Parsing an input text file
String inputTextFile = "Obama.txt";
String[] fileContents;
fileContents = loadStrings(inputTextFile);

fileContents has the source!

What next?
How do we turn fileContents into words?

- join array into one long string
  ```java
  String rawText;
  rawText = join(fileContents, " ");
  ```

- make all same case
  ```java
  rawText = rawText.toLowerCase();
  ```

- remove symbols and split string into words
  ```java
  String delimiters = " ,./?><;:'"[]\|\=+-()\*\^%$#@!~";
  tokens = splitTokens(rawText, delimiters);
  ```
Create a set of word frequency pairs.

Algorithm:
- create empty set pairs
- for each token
  - if pairs has (token, count)
    - increment count
  - otherwise
    - add (token, 1)
```java
class Word {
    // Each Word is a pair: the word, and its frequency
    String word;
    int freq;
    Word(String newWord) { // Constructor
        word = newWord;
        freq = 1;
    } // Word()
    String getWord() {
        return word;
    } // getWord()
    int getFreq() {
        return freq;
    } // getFreq()
    void incr() { // increments the word count
        freq++;
    } // incr()
    String toString() { // print representation of Word objects
        return "<"+word+", "+freq+">";
    }
} // class Word
```
Data Structures

- Ways of storing and organizing data

- Arrays
  - Must know the size ahead of time
  - Can not grow and shrink at will
Built-in Collection Classes

- **ArrayList**
  - A built-in object that stores and manages an *arbitrary* number of data items of any type (Objects).
  - Objects in an ArrayList are accessed by index \([0..\text{size}-1]\)

- **HashMap**
  - A built-in object that stores and manages an *arbitrary* number of data items of any type (Objects).
  - Objects in a HashMap are accessed by a **key**, which can be another Object, frequently a String.
ArrayList

- Constructors
  - ArrayList lst1 = new ArrayList();
  - ArrayList lst2 = new ArrayList(int initialSize);

- Fields

- Methods
  - size() // Returns the num of items held.
  - add(Object o) // Appends o to end.
  - add(int idx, Object o) // Inserts o at pos idx.
  - remove(int idx) // Removes item at pos idx.
  - get(int idx) // Gets items at idx. No removal.
  - set(int idx, Object o) // Replaces item at idx with o.
  - clear() // Removes all items.
  - isEmpty() // true if empty.
  - toArray() // returns an array that contains
    // the contents of the list
Make the set using an ArrayList

ArrayList<Word> wordFrequency = new ArrayList();

// Compute the wordFrequency table using tokens
for (String t : tokens) {
    // See if token t is already a known word
    int index = search(t, wordFrequency);
    if (index >= 0) {
        wordFrequency.get(index).incr();
    }
    else {
        wordFrequency.add(new Word(t));
    } // if
} // for
HashMap

- Constructors
  
  ```java
  HashMap map1 = new HashMap();
  HashMap map2 = new HashMap(int initialCapacity);
  ```

- Fields

- Methods
  
  ```java
  size() // Returns num of items held.
  put(Object key, Object o) // Puts o in map at key
  remove(Object key) // Remove Object at key
  get(Object key) // Get Object at key
  containsKey(Object key) // True if map contains key
  containsValue(Object val) // True if map contains val
  clear() // Removes all items.
  isEmpty() // true if empty.
  ```
Make the set using a HashMap?