CSE 1710

Lecture 18

Formatted Text

Goals/To do:

Given a string and a character, derive the **frequency of the character within the string**

Given a string, a target character and a replacement character, **implement character substitution.**

Given a numeric value in string format, **parse into numeric type**

Goals/To understand:

- difference between `char`, `String`, and `StringBuffer`
- The non-primitive String masquerades as a primitive type
- Pattern-matching abstractions (regular expressions)
- The difference between **raw** and **formatted** text; how to separate content from presentation

What is formatted text?

- plain text is text that does not contain styling information
- formatted test contains styling information
- styling information consists of:
  - font face aka font “name” (Times, Helvetica, Garamond, etc)
  - font colour
  - font style (plain/regular, italics, bold, italic and bold)
  - font size
- styling information can be used to introduce structure (look at this slide as an example)

How is text formatted?

- *body text* is annotated with formatting information
- these annotations are described as **mark up tags**
- the marked-up text is **rendered** in an environment that can process the mark-up tags
  - such environments include:
    - web browsers
    - GUI widgets
- A markup language is a modern system for annotating a text in a way that is syntactically distinguishable from that text
- how to keep the mark up tags distinct from the body text?
- another example of type-token distinction

Why do I want to format text using Java?

- control over presentation of text in GUIs
- server-side programming

What is html?

The source for this material is:
http://www.w3schools.com/html/

- HTML stands for Hyper Text Markup Language
- HTML is not a programming language, it is a markup language
- A markup language is a set of markup tags
- HTML uses markup tags to describe web pages; HTML documents are also called web pages

- HTML markup tags are usually called HTML tags
- HTML tags are keywords surrounded by angle brackets like <html>
- HTML tags normally come in pairs like and
- The first tag in a pair is the start tag, the second tag is the end tag
- Start and end tags are also called opening tags and closing tags

Example

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

L18Eg1.html

- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between and is displayed as a paragraph

- It does not matter how you format this html (line breaks, spaces), it will always be rendered the same way

L18Eg2.html

Example

- The example can be displayed using a web browser
- Introduce the class `HTMLTextDisplayer`

L18App1,
L18App2

Example

-
 is the tag for line break
- <hr /> is the tag for a horizontal (ruler)line
  - it is an empty element
  - it is self-closing

L18App3

HTML Element Syntax

- An HTML element starts with a start tag
- An HTML element ends with an end tag
- The element content is everything between the start and the end tag
- Some HTML elements have empty content
- Empty elements are closed in the start tag
- Most HTML elements can have attributes

- for comments:
  - start tag is "<!--"
  - end tag is "-->"

Text Formatting Tags

HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

L18App3

- Elements may be rendered differently depending on the context

L18App4

Style Attributes

- style attributes can be specified within an element's start tag to style the element
- E.g., L18App5
  - style="font-family:courier"
  - style="font-family:verdana;font-size:110%;color:green" L18Eg5.html
- *But this is not a very consistent way to style elements*
- Better way
  - Cascading Style Sheets (CSS)
  - a language for describing **presentation semantics**
- Three ways to do this:
  - in separate style sheet files (CSS files) **PREFERRED WAY
  - in the style element in the HTML head section
    - aka the "Internal" style sheet
  - in the style attribute in single HTML elements
    - aka the "inline" style

Colour Options (not exhaustive)

- Named
  - There are 147 predefined colour names
  - http://www.w3schools.com/cssref/css_colornames.asp
- Hexadecimal:
  - A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color. All values must be between 0 and FF.
  - e.g., color:#ff0000; /* this is red */
- RGB
  - specify the RGB values separately, either as number [0,255] or as percentage [0%...100%]
  - E.g.
    - color:rgb(255,0,0);
    - color:rgb(50%, 50%, 50%);

Font Face Options (not exhaustive)

Font Faces:

- In CSS, there are two types of font family names:
  - by generic family
    - a group of font families with a similar look (like "Serif" or "Monospace")
  - by font family
    - a specific font family (like "Times New Roman" or "Arial")
    - If the name of a font family is more than one word, it must be in quotation marks.
    - Eg.
      - font-family: "Times New Roman"

Font Styles:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Size Options (not exhaustive)

In CSS, font size can be specified either as absolute size or relative size.

- Absolute sets the text to a specified size
  - does not allow a user to change the text size in all browsers
  - bad for accessibility reasons
  - units: px, em (1em is equal to the current font size)
  - E.g.,
    font-size:2.5em;
    font-size:30px;
- Relative
  - use %
  - E.g.,
    font-size:100%;

The source for this material is:
http://www.w3schools.com/css/

Style Elements in the Head section

```
<html>
<head>
<style type="text/css">
body {background-color:yellow}
p {color:blue}
h1 {font-family:verdana;font-size:110%;color:green}
</style>
</head>
<body>
<h1>Here is a Heading</h1>
<p>Here is a basic paragraph of text.</p>
</body>
</html>
```

L18App6

The source for this material is:
http://www.w3schools.com/css/

User-Defined Styles

```
<html>
<head>
<style type="text/css">
body {background-color:yellow}
p {color:blue}
h1 {font-family:verdana;font-size:110%;color:green}
.mypara1 { color:black;font-family:serif }
</style>
</head>
<body>
<h1>Here is a Heading</h1>
<p>Here is a basic paragraph of text.</p>
<p class="mypara1">Here is a basic paragraph of text.</p>
</body>
</html>
```

L18App7

What can I do now?

1. Given plain text, format it in a certain way and display it
   - e.g., prompt the user for their first and family name, then prompt them for their middle name. Print out their full name in the order of first, middle, family name, each one in a different colour (red, green, blue)
   - you will do this following *good programming style conventions*
2. Given plain text, format it in a certain way and generate an html file.
   - e.g., same as above, but put output into a file instead of passing as arg to HTMLTextDisplayer
3. Given formatted text, modify its format
   - e.g., change all green text to blue

Next Lecture

We will review the answers to the review questions for Chapter 6.

If enough time, do selective insertion