Review

- Single Pixel Filters
  - Thresholding
  - Posterize
  - Histogram Equalization
  - Negative
  - Sepia
  - Grayscale
- Spatial Filters
  - Smooth
  - Blur – Low Pass Filter
  - Sharpen – High Pass Filter
  - Erosion
  - Dilation
- Image Processing Applications

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!";
```

msg

0	1	2	3	4	5	6	7	8	9	10	11	12	13
'I'	' '	'L'	'o'	'v'	'e'	' '	'C'	'S'	' '	'1'	'1'	'0'	'!'

Primitive Data Types

Type	Range	Default	Bytes
boolean	{ true, false }	false	?
byte	{ 0..255 }	0	1
int	{ -2,147,483,648 .. 2,147,483,647 }	0	4
long	{ -9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807 }	0	8
float	{ -3.40282347E+38 .. 3.40282347E+38 }	0.0	4
double	*much larger/smaller*	0.0	8
color	{ #00000000 .. #FFFFFFFF }	*black*	4
char	*a single character* 'a', 'b', …	'\u0000'	2

Making Strings

- Declaring String objects with no chars

```
String myName;
String myName = new String();
```

- Declaring String objects init'd w/ char array

```
String myName = "Fred";
String myName = new String("Fred");
```

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

Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec
(nul)	0	(dc4)	20	(	40	<	60	P	80	d	100	x	120
(soh)	1	(nak)	21	)	41	=	61	Q	81	e	101	y	121
(stx)	2	(syn)	22	*	42	>	62	R	82	f	102	z	122
(etx)	3	(etb)	23	+	43	?	63	S	83	g	103	{	123
(eot)	4	(can)	24	,	44	@	64	T	84	h	104	|	124
(enq)	5	(em)	25	-	45	A	65	U	85	i	105	}	125
(ack)	6	(sub)	26	.	46	B	66	V	86	j	106	~	126
(bel)	7	(esc)	27	/	47	C	67	W	87	k	107	(del)	127
(bs)	8	(fs)	28	**0**	**48**	D	68	X	88	l	108		
(ht)	9	(gs)	29	**1**	**49**	E	69	Y	89	m	109		
(nl)	10	(rs)	30	**2**	**50**	F	70	Z	90	n	110		
(vt)	11	(us)	31	**3**	**51**	G	71	[	91	o	111		
(np)	12	(sp)	32	**4**	**52**	H	72	\	92	p	112		
(cr)	13	!	33	**5**	**53**	I	73	]	93	q	113		
(so)	14	"	34	**6**	**54**	J	74	^	94	r	114		
(si)	15	#	35	**7**	**55**	K	75	_	95	s	115		
(dle)	16	$	36	**8**	**56**	L	76	`	96	t	116		
(dc1)	17	%	37	9	57	M	77	a	97	u	117		
(dc2)	18	&	38	:	58	N	78	b	98	v	118		
(dc3)	19	'	39	;	59	O	79	c	99	w	119		

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

Try it!

```
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()

Returns	Description
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()

Returns	Description
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);
  println(myByte);

}
```

Building Strings – Use '+'

```
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)

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

void setup() {

  tokens[0] = "one";
  tokens[1] = "two";
  tokens[2] = "three";
  tokens[3] = "four";
  tokens[4] = "five";

  println(tokens);
}
```

```
[0] "one"
[1] "two"
[2] "three"
[3] "four"
[4] "five"
```

Strings can be held by Arrays

– Initialized when declared

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

void setup() {
  println(tokens);
}
```

```
[0] "one"
[1] "two"
[2] "three"
[3] "four"
[4] "five"
```

Strings can be held by Arrays

– Not initialized

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

void setup() {
  println(tokens);
}
```

```
[0] null
[1] null
[2] null
[3] null
[4] null
```

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 );
}
```

```
[0] "12"
[1] " 34"
[2] " 56"
```

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

void setup() {
  as = splitTokens(s1, ":,");
  //as = trim(as);
  println( as );
}
```

```
[0] "Data"
[1] " 12"
[2] " 34"
[3] " 56"
```

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

```
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."
```

Remove spaces from a String

```
String stripSpaces( String s )
{
  for (int i=s.length()-1; i>=0; i--) {
    if (s.charAt(i) == ' ') {
      s = s.substring(0, i) + s.substring(i+1);
    }
  }
  return s;
}
```