Basic Computation

The Class String

THE STRING CLASS

Java API: String
(just like System & Math, only this time it has methods for working with character strings)

The String Class uses the String object.

Example:

```
String word; //declaring a String object
//note that the 'S' in String is U.C.
word = "Welcome!";
//double quotes used to encapsulate the literal
//initializing the identifier "word" to store the value "Welcome"

String greeting = "Hello World!";
//object declaration & initialization in one step
```

as opposed to the char data type…

```
char oneLetter = 'a'; //note the single quotes here
```

What values can be stored in a String?

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ASCII Code + the rest of the Unicode Character Set

http://www.unicode.org/charts/

The Empty String

- A string with no characters
- “” //adjacent double quotes
- “ ” //not the empty string - contains the space character

Reminder: When copying code and pasting into the IDE (jGrasp)
You may need to replace/retype the quote marks.
“ ”

More details on the String Class visit:

http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

String Storage

- *T*
  - Obje
- *ies of*
- cter
- tring
- ro (0).
- e String

Shhhh….here's a secret.

Using the String object
Data hiding is accomplished
Because the String is actually
stored as a
char ARRAY

Other Useful Methods of the String Object

String phrase = “Java is fun.”;

J	a	v	a		i	s		f	u	n	.
0	1	2	3	4	5	6	7	8	9	10	11

```
char a = phrase.charAt(2); //a = ‘v’
int b = phrase.length( ); //b = 12
//finding the first instance of a substring:
int c = phrase.indexOf( “fun”); //c = 8
```

Comparing String Objects

using String methods

Let's take a look at an example:
StringCompareDemo

method	syntax	return value(s)
equals	val1.equals(val2)	
compareTo	val1.compareTo(val2) < 0 val1.compareTo(val2) == 0 val1.compareTo(val2) > 0	
equalsIgnoreCase	val1.equalsIgnoreCase(val2)	
compareToIgnoreCase	val1.compareToIgnoreCase(val2) < 0 val1.compareToIgnoreCase (val2) == 0 val1.compareToIgnoreCase (val2) > 0	

Methods have a return type…

ex. main has a return type of void (no value is returned)

The methods of the String Object
return
boolean & int values

Let's see…

Concatenation of Strings

- joining 2 (or more strings together) to obtain a larger string

- + //the String Concatenation Operator (when used with Strings)
- the '+' operator must be placed between each String/variable identifier it with join
- other data types/literals, when concatenated with the String Concatenation Operator in an output statement, are converted to String data.

- substring(start,end) // using a portion of a string

String a = "This is work"; //'w' is at index 8

String b = "fun!"

String c = a.substring(0,8) + b; //c = This is fun!

T	h	i	s		i	s		w	o	r	k
0	1	2	3	4	5	6	7	8	9	10	11

Stay Inbounds!

- Remember!
  - The first value in a String is stored at index zero (0)
  - Thus, the last character in the String is at index (length -1)

If you try to reference an invalid
(out of bounds) index
you will get a Run-Time Error!

Info that can be part of a “string literal”:

The Escape Sequence!
(It all starts with a backslash)

- \n advances cursor to newline
- \t horizontal tab
- \b backspace
- \r return to beginning of current line
- \\ causes a backslash to be printed
- \’ causes a single quote to be printed
- \” causes a double quote

Note: The backslash does not count in the String length - it is considered invisible.

Basic Computation

Keyboard and Screen I/O

(I/O:input and output)

déjà vu review

Using print & println

For Screen OUTPUT

- print: displays output to standard output device (screen) but does not advance the cursor to a new line

- println: same as above, EXCEPT it does advance cursor to next line for future output commands

Info that can be part of the argument:

- “A string literal”
- myNum (variable value converted to string by …)
- + concatenate operator (used to connect strings)
  - Note: when concatenating strings watchyourwhitespace.

Keyboard Input

Utilities Package: java.util

Not all applications require keyboard input, thus, when keyboard input is needed - it is time to import a utility : the scanner for data entry .

```
import java.util.Scanner;
```

```
import java.util.Scanner;

public class InputScannerDemo
{

public static void main(String args[ ])
      {
          int number;
           Scanner keyboard = new Scanner(System.in);
           //declare variable(storage) of scanner type
           //identifier: keyboard (programmer defined)
           //keyword: new - create an object in memory (instantiate it)of Scanner type
           //(argument: type of scanner) System.in - standard concole input device - keyboard
           // = assigns the address of the object to keyboard
          System.out.print("Please enter an integer variable:  "); //This is a user prompt
          number = keyboard.nextInt();
          //nextInt method of Scanner class converts byte values to int data type
          // see page 85 for more Scanner class methods
          System.out.print("Your number was:  " + number);

       } //closing main method
} //closing class header
```

Scanner Class Methods

http://download.oracle.com/javase/1,5.0/docs/api/java/util/Sc

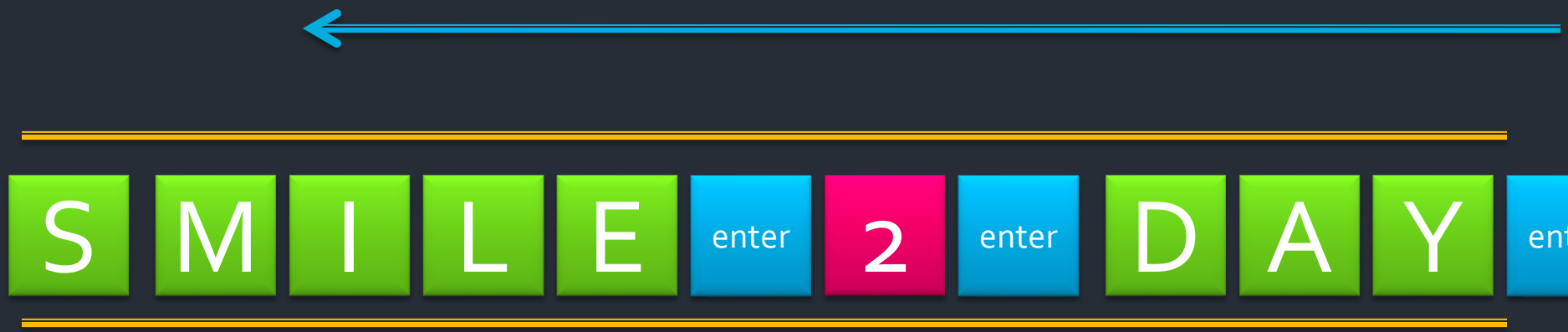
Delimiter: a separator of data
Sentinel: a character not found in the data

String	**next**() Finds and returns the next complete token from this scanner.
String	**next**(Pattern pattern) Returns the next token if it matches the specified pattern.
String	**next**(String pattern) Returns the next token if it matches the pattern constructed from the specified string.
BigDecimal	**nextBigDecimal**() Scans the next token of the input as a BigDecimal.
BigInteger	**nextBigInteger**() Scans the next token of the input as a BigInteger.
BigInteger	**nextBigInteger**(int radix) Scans the next token of the input as a BigInteger.
boolean	**nextBoolean**() Scans the next token of the input into a boolean value and returns that value.
byte	**nextByte**() Scans the next token of the input as a byte.
byte	**nextByte**(int radix) Scans the next token of the input as a byte.
double	**nextDouble**() Scans the next token of the input as a double.
float	**nextFloat**() Scans the next token of the input as a float.
int	**nextInt**() Scans the next token of the input as an int.
int	**nextInt**(int radix) Scans the next token of the input as an int.
String	**nextLine**() Advances this scanner past the current line and returns the input that was skipped.
long	**nextLong**() Scans the next token of the input as a long.
long	**nextLong**(int radix) Scans the next token of the input as a long.
short	**nextShort**() Scans the next token of the input as a short.

delimiter: ws

radix: number base of the value

The Keyboard input stream




```
x = keyboard.nextLine( );

y = keyboard.nextInt( );

z = keyboard.nextLine ( );
```

The Stream Fix for char data following numeric data:

keyboard.nextLine ( );

//not stored just consumed

```
 ----jGRASP exec: java InputScannerSampleMethods

Please enter an integer variable:  23
Your number was:  23Are you having fun, yet? (Yes or No)Yes
Ytext end Watch your white space so output does not run together!Please enter three words separated by a space: one two three
1st word: one
2nd word: two
3rd word: three
Please enter three words separated by the !: four and twenty!five!six!
1st word:
four and twenty
2nd word: five
3rd word: six

 ----jGRASP: operation complete.
```

```
            System.out.print("Please enter an integer variable:  "); //This is a user prompt
            number = keyboard.nextInt(); //nextInt method of Scanner class converts byte values to int data type
            System.out.print("Your number was:  " + number);

            keyboard.nextLine( );

            System.out.print ("Are you having fun, yet? (Yes or No)");
            word = keyboard.nextLine( );
            letter = word.charAt(0);
            System.out.print(letter);
            System.out.print("text end");
            System.out.print(" Watch your white space so output does not run together!");

            System.out.print ("Please enter three words separated by a space: ");
            word = keyboard.next();
            System.out.println("1st word: " + word);
            word = keyboard.next();
            System.out.println("2nd word: " + word);
             word = keyboard.next();
            System.out.println("3rd word: " + word);

            keyboard.useDelimiter("!");
            System.out.print ("Please enter three words separated by the !: ");
            word = keyboard.next();
            System.out.println("1st word: " + word);
            word = keyboard.next();
            System.out.println("2nd word: " + word);
             word = keyboard.next();
            System.out.println("3rd word: " + word);

        } //closing main method
} //closing class header
```

This next line is used to "eat" the newline character

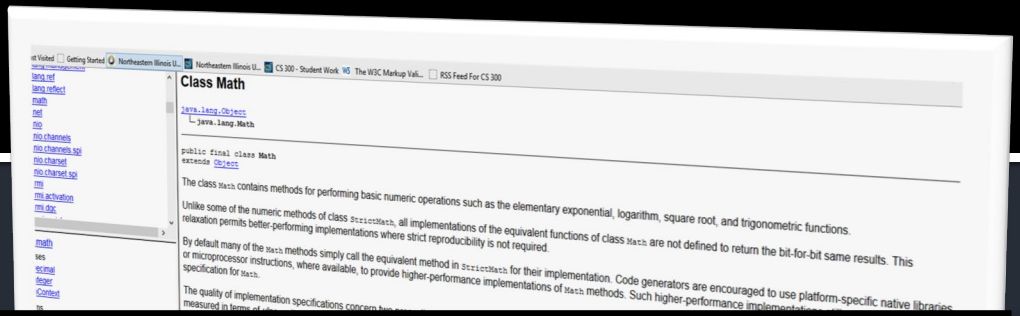
Reading a single char (example: Y (yes) or N (no)

```
:
String  input;
char answer;
System.out.print ("Are you having fun, yet?
                                    Y=yes, N=no)");
input = keyboard.nextLine( );
answer = input.charAt(0);
:
```

More on this concept when we get to arrays
Remember: indexing starts at Zero(0)

Math API

- http://docs.oracle.com/javase/1.5.0/docs/api/



Methods/Functions to know:

- pow
- sqrt
- ceil
- floor
- round
- max
- min
- abs
- random

Field Summary

static double	E The double value that is closer than any other to *e*, the base of the natural logarithms.
static double	PI The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

Method Summary

static double	abs(double a) Returns the absolute value of a double value.
static float	abs(float a) Returns the absolute value of a float value.
static int	abs(int a) Returns the absolute value of an int value.
static long	abs(long a) Returns the absolute value of a long value.
static double	acos(double a) Returns the arc cosine of an angle, in the range of 0.0 through *pi*.

static double	min(double a, double b) Returns the smaller of two double values.
static float	min(float a, float b) Returns the smaller of two float values.
static int	min(int a, int b) Returns the smaller of two int values.
static long	min(long a, long b) Returns the smaller of two long values.

ument and is equal to a mathematical integer.

Returns e^x -1.

static double	floor(double a) Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

polymorphism

Housekeeping: Decimal format Class

Display:

double data type: up to **15** decimal places

float data type: up to **6** decimal places

1. import java.text.DecimalFormat;
2. create an instance of Decimal format object
3. determine formatting template
   - # (wild card)
   - 0 (zero place holder)
   - , (comma)
   - $ (currency symbol)
   - % (converts value (*100) and displays percent symbol)
4. use template in output statement :
   System.out.println(**formatter1**.format(x)); **//identifier**

```
DecimalFormat formatter1 = new DecimalFormat("$#.00");
```

Let's take a look at an example: Decimal Format Class Example

Housekeeping: formatted display

printf method (printf Vs. println) code sample

Syntax: System.out.printf (FormatString, ArgumentList);

```
public class PrintfDemo
{
public static void main(String args[ ])
    {
      String u = "Pizza contains the 4 major food groups.";
       char w ='1'; //character 1 not numerical value 1
       int x = 3;
       float y = 32.7F;
       double z = 123456789.987654321;

        System.out.printf("I love pizza!\n");
        System.out.printf("Another reason to eat pizza: %s \n", u);
        System.out.printf("Value in w is: %c \n", w);
        System.out.printf("value in x is: %d, value in y is: %.2f, value in z is: %,.8f \n", x, y, z);

        System.out.printf("%10d \n",x);
        System.out.printf("%10.0f \n",y);
        System.out.printf("%10.0f \n",z);

    } //closing main method
} //closing class header
```

```
I love pizza!
Another reason to eat pizza: Pizza contains the 4 major food groups.
Value in w is: 1
value in x is: 3, value in y is: 32.70, value in z is: 123,456,789.98765433
         3
        33
 123456790
```

need to know largest possible value

Lab Time!

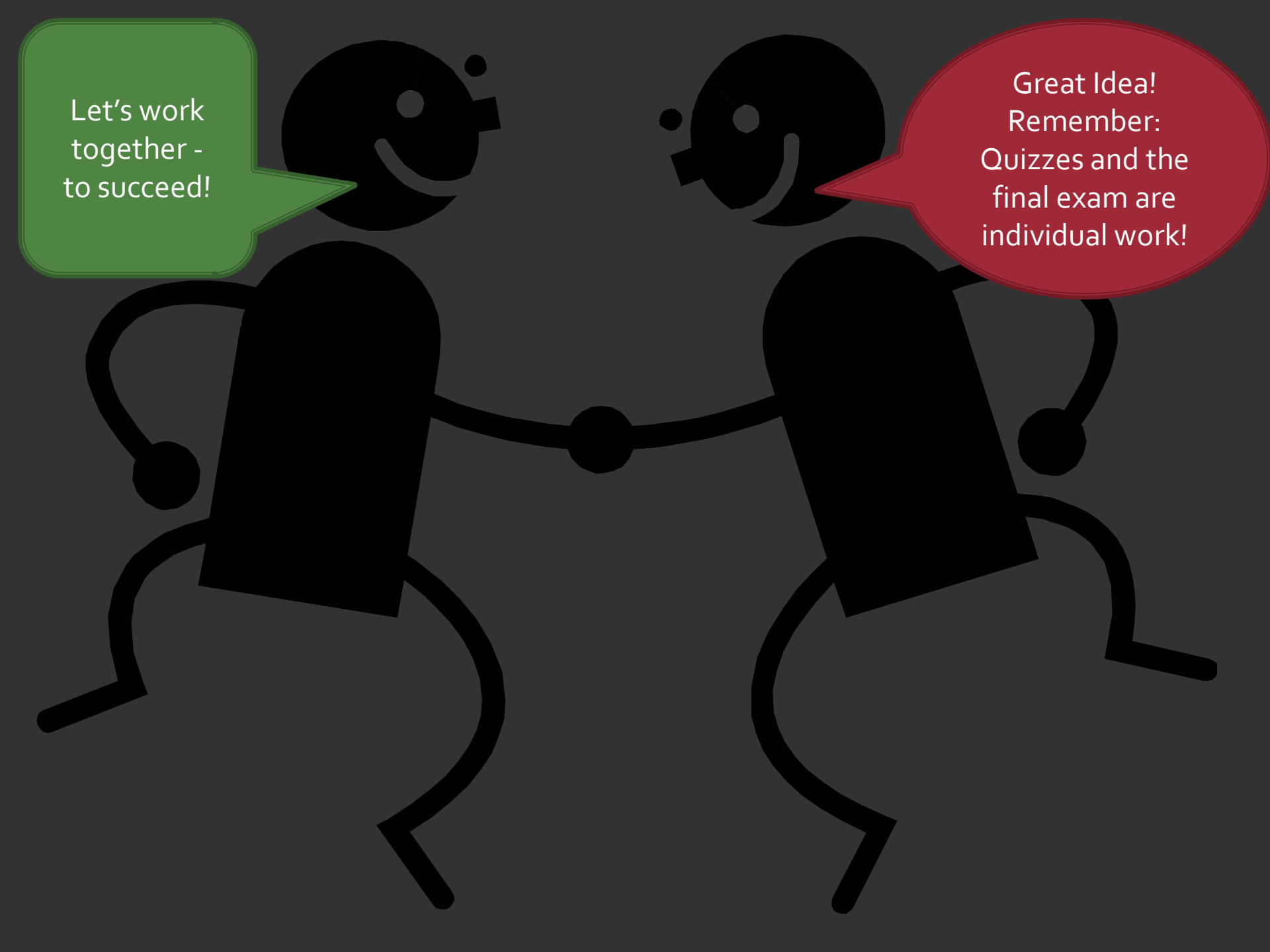
- Modify the most recent version of the payroll program:
  - allow for fractional portion of hours worked (i.e. 35.75 hours)
  - to have $ formatted output for the results, with only 2 digits to the right of the decimal place.

  (Print results twice, first using printf, then using decimal formatter object.)

Input Values:

Hours: 43.75 Rate: $250.25

Ans: $10,948.4375


Let's work together - to succeed!
Great Idea! Remember: Quizzes and the final exam are individual work!