AP Computer Science   Name: ______________________________
Chapter 2 Notes (6)

`switch` Statement, `char` and `String`s

Simple `String` Operations:

Concatenation: Connecting two Strings together. The addition of two `String`s.

Example:

```
String mm = "Hello";
String nx = "good buddy";
String c = mm + nm;
System.out.println(c);   // prints Hellogood buddy
```

or

```
System.out.println("Hello" + "  good buddy");
          // prints Hello good buddy
```

The `length()` method:

Use the `length()` method to find the number of characters in a `String`:

```
String someName = "Donald Trump";
int length = someName.length();
System.out.println(length);
     // prints 12...notice the space gets counted
```

Right now we don't see much value in this length method, but just wait!

A piece of a `String` (substring):

We can pick out a piece of a `String`...substring

Example:

```
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4);
System.out.println(smallPart);  // prinks ky the dog
```

Why do we get this result? The various characters in a `String` are numbered starting on the left with 0. These numbers are called **indices**. (Notice the spaces are numbered too.)

S	***p***	***a***	***r***	***k***	***y***		***t***	***h***	***e***		***d***	***o***	***g***
0	1	2	3	4	5	6	7	8	9	10	11	12	13

A more useful form of substring:

There's another way to use substring. The method is overloaded.

```
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4, 12);
System.out.println(smallPart);  // prints ky the d
```

S	***p***	***a***	***r***	***k***	***y***		***t***	***h***	***e***		***d***	***o***	***g***
0	1	2	3	4	5	6	7	8	9	10	11	12	13

What does `myPet.substring(4, 12);` do?

Conversion between lower and upper case:

`toLowerCase()` converts all characters to lower case (small letters)

```
String bismark = "Dude, where's MY car?";
System.out.println(bismark.toLowerCase());
          // prints dude, where's my car?
```

`toUpperCase()` converts all characters to upper case (capital letters)

```
System.out.println("Dude, where's MY car?".toUpperCase());
          // prints DUDE, WHERE'S MY CAR?
```

Concatenating a `String` and a numeric:

It is possible to concatenate a String with a numeric variable as follows:

```
int x = 27;
String s = "Was haben wir gemacht?";
          // German for "What have we done?

String combo = s + " " + x;
System.out.println(combo);
          // prints Was haben wir gemacht? 27
```

Escape sequences:

How do we force a **quote** character (") to printout...or, to be part of a String. Use the **escape sequence**, \", to print the following (note: escape sequence always starts with the \ character.

Example:

```
String s = "What \"is\" the right way?";
System.out.println(s);

// prints What "is" the right way?
```

Other Escape Sequences:

Desired Character	Escape Sequence	Meaning
	\b	backspace
	\t	tab
	\n	new line (also called line break)
	\r	carriage-return
	\f	form feed
“	\”	double quotation mark
‘	\’	single quotation mark
\	\\	backslash

Example:

```
String s = "Here is one line\nand here is another.";
System.out.println(s);
```

Prints:

```
Here is one line
and here is another.
```

Example:

```
System.out.println("Path = c:\\nerd_file.doc");
```

Prints:

```
Path = c:\nerd_file.doc
```

Example:

```
System.out.println("Name:\t\tAddress:");
```

Prints:

```
Name:           Address:
```

`switch` Statement:

The `if` statement is the most powerful and often used decision-type command. The switch statement is useful when we have an integer variable that can be one of several quantities.

For example, consider the following **menu** scenario (enter and run this program):

```
//This code should be place inside the main method of a class

System.out.println("  1. Addition");
System.out.println("  2. Subtraction");
System.out.println("  3. Multiplication");
System.out.println("  4. Division\n");

System.out.print("  Your Choice? ");

Scanner reader = new Scanner(System.in);
int choice = reader.nextInt();

System.out.print("\nEnter first operand: ");
double op1 = reader.nextDouble();

System.out.print("\nEnter second operand: ");
double op2 = reader.nextDouble();

System.out.println("");

switch(choice)

{
case 1: //addition
     System.out.println(op1 + “ plus ” + op2 + “ = ” + (op1 + op2) );
     break;
case 2: //subtraction
     System.out.println(op1 + “ minus ” + op2 + “ = ” + (op1 – op2) );
     break;
case 3: //multiplication
     System.out.println(op1 + “ times ” + op2 + “ = ” + (op1 * op2) );
     break;
case 4: //division
     System.out.println(op1 + “ divided by ” + op2 + “ = ” + (op1 / op2) );
     break;
default:
     System.out.println(“Hey dummy, enter only a 1, 2, 3, or 4!”);
}
```

The optional `default`:

The `default` command is optional. You can you it if there might be a possibility of the value of choice not being one of the cases.

Give me a `break`:

The `break` statements are normally used. Try leaving them out and see what happens here. In the next section we will look at an application in which they are omitted.

Basically, `break` jumps us out of the `switch` structure and then code execution continues with the first line immediately after the closing `switch` brace. Specifically, you might want to omit the `break` within the *case 1: section*. If choice is *1* then the result will be that it prints the answer for **both** addition and subtraction.

The next experiment you might want to do is to leave the parenthesis off of `(op1 + op2)` in the *case 1: section*. Since `op1 + op2` is no longer in parenthesis, the plus between them no longer means addition. It now means concatenation since all the activity to the left of this point in the code was also `String` concatenation.

Leaving off the break:

Now, let's look at an example where we intentionally omit `break`:

```
// Suppose at this point in the program we have an integer variable, j.
// If j equals 1, 2, or 3 we want to set String variable s to “low” and if
// j equals 4, 5, or 6 we want to set s to “high”. If j equals 7, set s to
// “lucky”.

switch ( j )
{
     case 1:
     case 2:
     case 3:
          s = “low”;
          break;
     case 4:
     case 5:
     case 6:
          s = “high”;
          break;
     case 7:
          s = “lucky”;
}
```

A new data type… `char`:

Before we look further at the `switch` statement, we must look at a new data type, `char`. This stands for character. Following is a typical way to declare and initialize a character:

```
      char ch = ‘h’;
```

Notice that a character is always enclosed in single quotes. Characters can be anything, even numbers or symbols:

```
      char x = ‘6’; char pp = ‘@’;
```

`int` and `char` are permissible types:

`switch( )` statements primarily switch on integers or characters (`short` and `byte` types can also be used, but rarely are). Modify the example on the previous page to `switch` on a `char` instead of int. See the next page for the necessary modifications:

```
System.out.println(“Make your arithmetic selection from the choices below:\n”);
System.out.println(“ A. Addition”);
System.out.println(“ S. Subtraction”);
System.out.println(“ M. Multiplication”);
System.out.println(“ D. Division\n”);

System.out.print(“ Your choice? ”);

Scanner reader = new Scanner(System.in);
String choice = reader.nextLine( );
//char ch = choice; //You would think this would work…but it doesn’t.
char ch = choice.charAt(0); //you just learned another String method.

System.out.print(“\nEnter first operand. ” );
double op1 = reader.nextDouble( );
System.out.print(“\nEnter second operand .” );
double op2 = reader.nextDouble( );
System.out.println(“ ”);

switch (ch)
{
case ‘A’: //addition
case ‘a’: //Notice we are providing for both capital A and little a.
      System.out.println(op1 + “ plus ” + op2 + “ = ” + (op1 + op2) );
      break;
case ‘S’: //subtraction
case ‘s’:
      System.out.println(op1 + “ minus ” + op2 + “ = ” + (op1 - op2) );
      break;
case ‘M’: //multiplication
case ‘m’:
      System.out.println(op1 + “ times ” + op2 + “ = ” + (op1 * op2) );
      break;
case ‘D’: //division
case ‘d’:
      System.out.println(op1 + “ divided by ” + op2 + “ = ” + (op1 / op2) );
      break;
default:
      System.out.println(“Hey dummy, enter only a A, S, M, or D!”);
}
```