Immediate if or the Conditional Operator in Java

```
Scanner input = new Scanner(System.in) ;

System.out.print( "Enter a number:"); int x = input.nextInt( );
input.nextLine( );

String someString = ( x > 10000) ? "a big number!" : "a relatively small number" ;

System.out.println(  "The computer thinks that you entered" + someString );
```

Now let's look at:

```
String someString = ( x > 10000) ? "a big number!" : "a relatively small number" ;
```

( ) is Java syntax of a boolean conditional statement

? is Java syntax which serves as "then" in an if/then clause

: is Java syntax which serves as "else" in an if/then clause.

Immediate if or the Conditional Operator in Java

```
String someString = ( x > 10000) ? "a big number!" : "a relatively small number" ;
```

( ) is Java syntax of a boolean conditional statement

? is Java syntax which serves as "then" in an if/then clause

: is Java syntax which serves as "else" in an if/then clause.

The statement/object/variable after the ? is returned if the boolean is true

The statement/object/variable after the : is returned if the boolean is false

This syntax is a very concise syntax omitting any Java (key) words. To read this in your mind think aloud the green words which is Sakas-talk, not Java!!!!

```
String someString = IF ( x > 10000) THEN RETURN "a big number!"
                    ELSE RETURN "a relatively small number" ;
```

Bottle or Bottles: One solution

```
String bot1 = ( n == 1 )     ? "bottle"  : "bottles" ;
String bot2 = ( n-1 == 0 ) ? "bottles" : "bottle" ;

System.out.println (
        n +  " " + bot1 +    " of beer on the wall,\n" +
        n + " " + bot1 +    " of beer,\n
       Ya' take one down, ya' pass it around,\n" +
       (n-1) +"  " + bot2 + " of beer on the wall."
);
```

Bottle or Bottles: More elegant solution

Wrap the conditional operator in a function

```
public static String pluralOrNot ( int numBottles )  {

    return ( numberBottles == 1 )  ? "bottle" : "bottles";

}


  System.out.println (
          n +  " " + pluralOrNot( n ) +    " of beer on the wall,\n" +
          n + " "  + pluralOrNot( n ) +    " of beer,\n
          Ya' take one down, ya' pass it around,\n" +
         (n-1) +"  " + pluralOrNot( n-1 ) + " of beer on the wall."
  );
```

Java strings: The String type

```
String firstName ;
firstName = “William ”;

String lastName = “Sakas”;

String middleName = new String(“Gregory ”);


System.out.println(firstName +middleName+lastName);


int gregNumOfChars = middleName.length();

System.out.println(gregNumOfChars); // 7
```

Three ways to create an initialize strings

There are also useful string functions, note the “dot” – the length function is a string METHOD – much more on this later.

Java String Concatenation

- Concatenation: Putting two strings together
- In Java we use the plus operator (+) to make strings grow.

```
String food = “ice”;
food = food + “ cream”;
System.out.println( food );
```

- The plus operator also words with character data

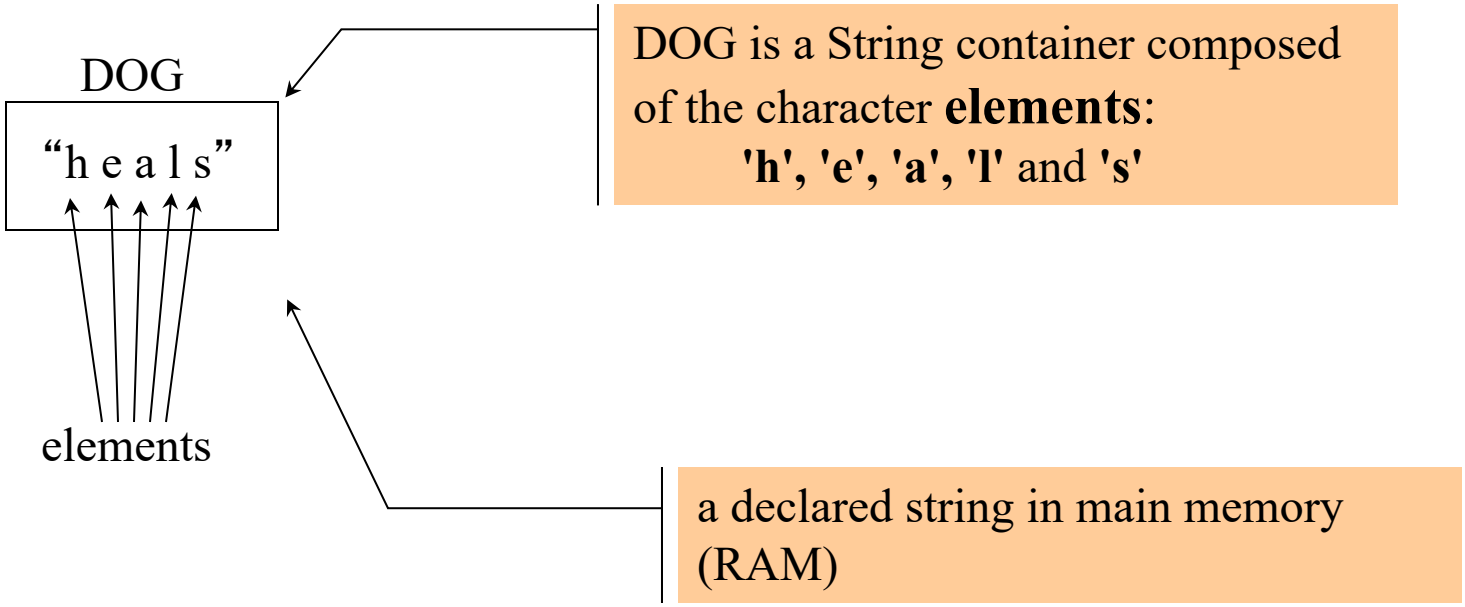
```
food = food + “ cone”;
food = food + 's';
food = 'l' + food;
```

- But one side has to be a string

```
String newString = ‘a’ + ‘b’; // SNYTAX ERROR!
```

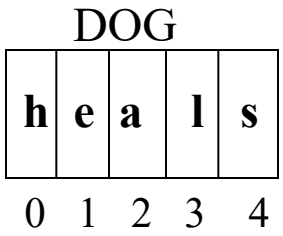
The String type is a ***container*** type - Strings contain a collection of sub-***elements*** which can have different values

String DOG = “heals” ;



Additional functionality provided by the string type:

"**in position read**" access



strings are made up of char **elements**
and each element has an **index** starting from 0.

in position READ access to element with index 2

```
System.out.println( DOG.charAt( 2 ) );
```

charAt member function is an *indexing or subscripting* function
In **your head**, interpret charAt as "in position" or "at position"

"output DOG IN POSITION 2."

"output DOG AT POSITION 2."

* experienced programmers often use "sub" when reading off the subscript operator.

DOG

h	e	e	l	s
0	1	2	3	4

Integer variables are often used to hold indices.

```
int  anInt = 1 ;
System.out.println( DOG. charAt (anInt) );
anInt++ ;
System.out.println( DOG.charAt ( anInt ) );
anInt++ ;
System.out.println( DOG.charAt ( anInt ) );
```

TASK: Write some Java code to output a string in reverse order.

precondition: aString = = "heels"
postcondition: **sleeh** is displayed on the screen.

Hint: Use an integer as an index. Which index do we start with?

Let's write some pseudocode.

1) initialize an index (theIndex) to the position of the last character

2) while theIndex is greater than or equal to zero:

    output the character in the position of theIndex.

    subtract one from theIndex.

Now the Java Code:

```
String aString = "heels" ;
int theIndex = aString . length() - 1 ;
while ( theIndex >= 0 )
{
        System.out.print( aString.charAt( theIndex ) ) ;
        theIndex-- ;
}
```

Containers and For loops

- **We as programmers must ALWAYS be able to access the size of a container.**
- With strings this is easy: **aString . length()**.

- So **For … Loops** are a natural way to process containers.

New C++ Code for displaying a string backwards:

```
string aString = "heels" ;

for (int theIndex = aString . length() - 1 ;
         theIndex >= 0 ;
         theIndex-- )
{
       System.out.println( aString.charAt( theIndex ) );
}
```

Containers and For loops

Casting - temporary change in data type.

Converting character to ASCII code:

```
char ch = 'Z';
System.out.println( (int) ch  ); // displays 90
```

Converting ASCII code to character:

```
int anInt = 90;
System.out.println( (char) anInt  ); // displays Z
```

```
// Displays the ascii code for all elements of a string
String someString = "feline";

for (int i=0; i<someString . length(); i++)
{
    System.out.print( (int) someString.charAt( i ) + "  " ) ;
}
```

Containers and For loops

// Converts a string: all lower case characters (ascii 97-122) are converted
// to upper case (ascii 65-90)

Write Pseudocode first:

0) Create a new empty string upperStr, which will be all upper case

1) Go through the characters (elements) of a string from beginning to end:

1b) get the ascii code of the current character

1c) if the ascii code of a character is between 97-122
subtract 32 from the character
create a new char which will be upper case
using the new ascii code
concatenate the new char to upperStr

else // not upper case

concatenate the current character to upperStr

```
// Converts a string: all lower case characters (ascii 97-122) to upper case (ascii 65-90)
String upperString = "";

String someString = “Edited Jul 10 at Twenty-Two hundred hours”;

// Go through the characters (elements) of a string from beginning to end:
for (int i=0; i< someString . length() ; i++)
{
        // get the ascii code of the current character
        int asciiCode = (int) someString.charAt( i ) ;

        // if the ascii code of a character is between 97-122
        if (asciiCode >= 97 and asciiCode <= 122)
        {
            // subtract 32 from the character
            upperAsciiCode = asciiCode – 32 ;

            //  convert the character to upper cases using the new ascii code
            //    and concatenate to upperString
            upperString = upperString + (char) upperAsciiCode ;
        }
        else upperString = upperString + someString[i] ; // or (char) asciiCode
}
System.out.println( upperString );
// Displays: EDITED JUL 10 AT TWENTY -TWO HUNDERED HOURS
```

Containers and WHILE loops

// Prints out the second word of a sentence entered by a user. You can assume that the
// sentence does not begin with spaces, and that words are separated by spaces only
// and there are at least three words in the sentence.

Write simple pseudocode first:

1) **Go through the first word character by character without displaying anything**
2) **Go through all the spaces after the first word**
3) **Go through the second word character by character and display each character**

Refine pseudocode:

1) **Go through the first word character by character without displaying anything**
   1a) Look for the first character that *is* a space.
2) **Go through all the spaces after the first word**
   2a) Look for the first character that *is not* a space
3) **Go through the second word character by character and display each character**
   3a) until a space is reached.

HW: Adapt code so that it works for one- or two-word sentences as well.

Containers and WHILE loops

```
// Prints out the second word of a sentence
 String someString = "Edited Jul 10 at Twenty-Two hundred hours.";

 // Go through the first word character by character without displaying anything
 // Precondition: someString does not begin with spaces. Words are separated by spaces.
 // Thus, someString in position 0  is not a space char.
 int i = 0;
 while ( someString . charAt( i ) != ' ')
 {
            i++ ;
 }
 // Postcondition: someString in position  i  IS a space.
 // Go through all the spaces after the first word
 while (someString . charAt( i ) == ' ')
 {
            i++ ;
 }
 // Postcondition: someString in position i  IS NOT a space.
 // Go through the second word character by character and display each character.
 // Precondition: someString[ i ] is the first letter of the second word
 while (someString [ i ] != ' ') // This only works for sentences of three words or more. Try to fix it!
 {
            System.out.print( someString.charAt ( i ) );
            i++ ;
 }
```