JAVA QUICK GUIDE : INPUT/OUTPUT – TO/FROM SCREEN

Output text that is not stored in a variable:

```
System.out.println ("Hello there, how are you");
```

Output text that is not stored in a variable, as well as variable values:

```
System.out.println ("Hello there, you are " + age + " years old");
```

Print out a blank line:

```
System.out.println ();
```

Set up your program so that it can receive input:

```
import java.util. Scanner;                          //you'll need to add this line
public class Main {                                 //this will already be there
public static void main(String[] args) {            //this will already be there
Scanner keyedInput = new Scanner(System.in);        //you'll need to add this line
double value;                                       //this just declares a variable
value = keyedInput.nextDouble();                    //this reads a value
```

Receive input on the same line as the prompt:

```
System.out.print ("Enter your age: ");
```

*Notice that in the above code print is used instead of println, this will stop the output from dropping down a line

Output a decimal number to a given number of decimal values:

```
System.out.format("%.5f%n", amount);
```

*The code above will output the value to 5 decimal numbers, you would just change the number 2 if you wanted to output the number to two decimal values. Note that amount would have been declared as type double.

Adding symbols before outputting variable values:

```
System.out.println("Your account balance is $" + acctBalance);
```

*Notice how the dollar sign is included in the line of text, so that it will appear just before the account balance is outputted.

Generating a random number:

```
value = Math.round(Math.random()*9 + 1);  //generate a random number between 1 and 10
```

JAVA QUICK GUIDE : LOOPS AND STRINGS

Creating a for loop from 1 to 10:

```
for (int i=1;i<=10;i=i+1)
{
System.out.println (i);
}
```

Creating a while loop:

```
int count;

count = 3;

while (count <= 7)
{
System.out.println ("Hello");
count = count + 1;
}
```

STRINGS:

.length Returns the number of characters in the string to a waiting variable. **lengthOfWord = inputWord.length();**	**.substring (start)** Returns a smaller string that includes the characters beginning at the number indicated and ending at the end of the string. **smallString = inputWord.substring (3);**
.substring (start, end) Returns a smaller string that includes the characters beginning and ending at the numbers indicated. **smallString = inputWord.substring (4,7);**	**.lastIndexOf ("a character");** Returns the last location where the search character was found. **location = inputWord.lastIndexOf("S");**
.charAt (location); Returns the character that is at the location indicated. **oneChar = inputWord.charAt (4);**	**.indexOf ("a character");** Returns the first location where the search character was found. **location = inputWord.indexOf ("a");**
.toUpperCase(); Returns the string in all upper case **capitals = inputWord.toUpperCase ();**	**.toLowerCase();** Returns the string in all lower case **lowers = inputWord.toLowerCase ();**

JAVA QUICK GUIDE : SELECTION AND ARRAYS

Selection:

```
if (testscore >= 90)
      {
         grade = 'A';
      }
else if (testscore >= 80)
      {
            grade = 'B';
      }
else if (testscore >= 70)
      {
            grade = 'C';
      }
else if (testscore >= 60)
      {
            grade = 'D';
      }
else
      {
            grade = 'F';
      }
```

Not equal	!=
Less than	<
Less than or equal to	<=
Greater than	>
Greater than or equal to	>=
Equal to	==
AND	&
OR	|
NOT	!

```
EX: if (roll ==7)|(roll==11)
        {
           STUFF GOES HERE
        }
```

Arrays in Java…

```
String [ ] names = new String [10];

names[0] = "Jim";              //notice that arrays in Java start at 0
names[1] = "Bob";
```

Using the array class to sort…

You need to import the Array class at the top of your program.

```
import java.util.Arrays;
```

Once your array is full, one line will sort it…

```
Arrays.sort(names)             //this will sort your array called names
```

Using the array class to search…

You need to import the Array class at the top of your program.

```
import java.util.Arrays;
```

You will need the names array to be filled and a search key (what you're looking for) and an int to hold its location.

```
int foundAt;
String lookingFor;

lookingFor = "john";

foundAt = Arrays.binarySearch (names, lookingFor);    //will search the names array
                                                      //to find lookingFor
```

JAVA QUICK GUIDE : SWITCH STATEMENTS, VARIABLE TYPES,

If… then… else statements can be replaced with SWITCH statements (similar to CASE statements in Turing)

```
switch (variable)
{
      case 1:
            System.out.println ("this is option 1");
            break;
      case 2:
            System.out.println ("this is option 2");
            break;
      default:
            System.out.println ("that's not an option");
}
```

JAVA DATA TYPES			
TYPE	**STORAGE**	**MAXIMUM**	**MINIMUM**
byte	8 bit Uses signed twos complement	127 Stored as: 01111111	-128 Stored as: 10000000
short	16 bit Uses signed twos complement	32 767 Stored as: 01111111 11111111	-32 768 Stored as: 10000000 00000000
int	32 bit Uses signed twos complement	2 147 483 647	-2 147 438 648
long	64 bit Uses signed twos complement	9 223 372 036 854 775 807	-9 223 372 036 854 775 808
float	32 bit Uses single precision 32-bit IEEEE 754 floating point Don't use for precise values, don't use for currency.	Humongous!	Negative Humongous!
double	64 bit Uses double precision 64-bit IEEEE 754 floating point Don't use for precise values, don't use for currency.	Extra Humongous!	Negative Extra Humongous!
boolean	Represents one bit (either 1 or 0)	TRUE	FALSE
char	16 bit Unicode character (converts the binary value into a character)	0	65 535

Where's the **string**? Strings are not a primitive data type in JAVA. JAVA uses a class to use strings.