Java:

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

C++:

```
#include <iostream>
#include <stdlib.h>  //technically you don’t need this library for this program, but
                     //for most programs you will need the standard library so it’s
                     //good to get into the habit of including it.
using namespace std;

int main() {
       cout <<"Hello world!" << endl;
       return (0);
}
```

Java Primitive Types:

byte	8 bits
short	usually a 16-bit int
int	usually a 32-bit int
long	usually a 64-bit int
float	floating decimal point number
double	double the size of a float
boolean	true/false
char	single unicode character

C++ Primitive types:

*	
int	usually a 32-bit int
*	
float	floating decimal point number
double	double the size of a float
bool	true/false
char	single unicode character

*Note that in C++, you can make chars and ints signed (the default) or unsigned. By making an int unsigned, you get an extra bit (and thus larger numbers are possible) but you can’t have a negative number.
You can also specify that the int is either short or long. If it is short, it uses 16 bits, whereas if it a default int, it would be 32 bits, whereas if it is a long int, it would be 64 bits.
Try the following:

```
    cout << "Size of char : " << sizeof(char) << " byte" << endl;
    cout << "Size of int : " << sizeof(int) << " bytes" << endl;
    cout << "Size of short int : " << sizeof(short int) << " bytes" << endl;
    cout << "Size of long int : " << sizeof(long int) << " bytes" << endl;
    cout << "Size of signed long int : " << sizeof(signed long int) << " bytes" <<
endl;
    cout << "Size of unsigned long int : " << sizeof(unsigned int) << " bytes" <<
endl;
    cout << "Size of float : " << sizeof(float) << " bytes" <<endl;
    cout << "Size of double : " << sizeof(double) << " bytes" << endl;
```

Java Operators:

postfix	x++,x--
Multiplication:	*
Division	/
Modulus:	%
Addition:	+
Subtraction	-
Relational:	>,<,>=,<=
Equality:	==,!=
Logical and:	&&
Logical Or:	||
Assignments:	=,+=,-=,*=,/=,

C++ Operators:

postfix	x++,x--
Multiplication:	*
Division	/
Modulus:	%
Addition:	+
Subtraction	-
Relational:	>,<,>=,<=
Equality:	==,!=
Logical and:	&&
Logical Or:	||
Assignments:	=,+=,-=,*=,/=,

Java if:

```
int testscore = 76;
char grade;

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';
}
System.out.println("Grade = " + grade);
```

C++ if:

```
int testscore = 76;
char grade;

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';
}
cout << "Grade = " << grade << endl;
```

Java Comments:	**C++ Comments:**
`/*  comments go`	`/*  comments go`
`    here`	`    here`
`*/`	`*/`
`Or`	`Or`
`//comments go here`	`//comments go here`

Java While Loop:	**C++ While Loop:**
`int count = 1;`	`int count = 1;`
`int tot = 0;`	`int tot = 0;`
`while (count < 11) {`	`while (count < 11) {`
`    tot += count;`	`    tot += count;`
`    count++;`	`    count++;`
`}`	`}`
`System.out.println(tot);`	`cout << tot << endl;`

Java Do-While Loop:	**C++ Do-While Loop:**
`int count = 1;`	`int count = 1;`
`int tot = 0;`	`int tot = 0;`
`do {`	`do {`
`    tot += count;`	`    tot += count;`
`    count++;`	`    count++;`
`} while (count < 11);`	`} while (count < 11);`
`System.out.println(tot);`	`cout << tot << endl;`

Java For Loop:

```
for(int i=1; i<11; i++){
    System.out.println("Count: "+i);
}
//NOTE: you can skip any of the 3
conditions if defined elsewhere
```

C++ For Loop:

```
for(int i=1; i<11; i++){
    cout << "Count: " << I << endl;
}
```
//NOTE: you can skip any of the 3 conditions if defined elsewhere

Java Arrays:

```
int[] anArray;
anArray = new int[5];
anArray[0] = 100;
anArray[1] = 200;
anArray[2] = 300;
anArray[3] = 400;
anArray[4] = 500;
```

OR

```
int[]   ints2 = new int[]{1,2,3,4,5};
```

Multidimensional:

```
int[][] intArray = new int[10][20];
```

C++ Arrays:

```
int c[5];
c[0]=100;
c[1]=200;
c[2]=300;
c[3]=400;
c[4]=500;
```

OR:

```
int a[5] = {3,2,4,1,7};
```

Multidimensional:

```
int marr[10][20];
```

you can do:

```
int marr2[3][4] = {
    {6,3,2},
    {8,1,3},
    {3,5,1},
    {7,8,9}     };
```

C++ Functions (Java doesn't have functions – it only has methods):

C++ requires function declarations before you use the function. So above the main function in your file (and below the include libraries) you should declare each function you will write and use. A function declaration consists of the return type, the name of the function, the number and type of the input parameters, and a semicolon; See below for an example.

There is another (usually better) way to do this. We'll discuss that later in the class. For now include all function declarations above the main function in your file.

Your file should look something like this (date and time don't have to look exactly like this – I know eclipse automatically inserts the date in a different way and that's fine):

```
/* Debra Yarrington
 * TA's name
 * 8/14/17
 * This file contains functions for lab 1.  The functions aren't necessarily related
 * in any way other than that they are required for lab 1.
*/

#include <iostream>
#include <stdlib.h>
using namespace std;

int func(int k[], int size) ;  //function declaration

int main() {
       …  //call functions here
       return(0);
}

/* This function takes as an input parameter an array of integers and an integer.  The second
parameter is the number of elements in the first parameter (the array)  Tue function sums the
integers in the array.  It then returns an int, the sum of the values in the array
int func(int k[], int size) { //function definition
       int a = 0;
       for (int i = 0; i < size; i++) {
              a += k[i];
       }
       return a;
}
```