1 Table of Contents

Reading this Document:

```
Code is represented in Blue.
```

Commands that you need to do are represented in Green.

Important Texts are represented in Orange.

2 Introduction To Java

The information in this section is only for reading, and interest only. You will not be marked on any of it in the programming section of Grade 10 IT.

Java is a programming language, originally developed in 1995 making it 26 years as of 2021[1].

To understand what a programming language is first, you need to understand what an operating system[2] is.

There are various operating systems, abbreviated as OS, currently in existence. Three mainstreams

OS's that you would have come across, are:

- Google Android running on majority of world's smartphones.
- Apple iOS also running on smartphones, proprietary software of Apple Inc.
- Microsoft Windows running on majority of the world's desktop systems.

A Programming language allows us to create software, that will run and have functionality on these operating systems. There are currently +- 700 programming languages in the world[3].

Java has consistently ranked in the top5 used programming languages since its inception. It could be argued that it the most popular programming language in the world.

Due to it's high popularity, there are a vast amount of resources available covering the Java Language. However it is always best practice, to get information from the source.

By this it is meant, that if you are seeking to learn Java, then you should visit Oracle (owners of Java) website, first to query information, until such a time you acquire your own reliable sources.

On a final note, a program written in Java on Windows 10, can then be ported without much changes in the code to Google Android and run there also. This is due to Java's (Write Once, run Anywhere (WORA)) concept.

[1]https://www.java.com/en/download/help/whatis_java.html

[2] https://www.tutorialspoint.com/operating_system/os_overview.htm

[3] https://en.wikipedia.org/wiki/List_of_programming_languages

3 SYNTAX

The word syntax refers to a programming language's rules.

Consider the English sentence, *"The man goes to the shop."* We cannot rephrase this sentence, as the *"The shop goes to the man"*

It is not valid/allowed in English. We can say that it has "broken" an English Syntax rule.

Similarly, in the Java Programming Language, there are Syntax rules that make the language "valid". Three of them that you should be immediately familiar with are :

1. Java sentences are case sensitive. Consider this valid Java statement which prints out "Hello Sumir"

```
 System.out.println("Hello Sumir");
```

Now if we change the p from lowercase to Uppercase in printLn as per below :

```
System.out.Println("Hello Sumir");
```

This will result in a Syntax Error, your entire program will not work. Even though you have spelt everything correctly, the CASE of statements is important. Not to worry though, errors are easily identified by Netbeans or JGrasp.

2. **Statements end in semicolons**. You can think of this one, as English has full stops at the end of each sentence, in Java we need a semicolon. The semicolon states that we have finished with this sentence.

```
System.out.println("Hello Sumir");
```

3. Brackets need to be matched. By this it is meant, if you have an opening bracket such as **(** or **{** you need a corresponding **)** or **}** respectively. This will be understood more clearly later, on, but for now just take note of it.

If any of these rules are violated, your program will not compile known, as a Syntax Error has been detected, and it is then your duty to go figure out what is wrong. This process of figuring out what the error is present in the code is known as Debugging.

4 Hello World

The Hello World Program has a long tradition in the programming world.

We will be working with jGRASP, to keep things simple at this point. Where do we start programming ? We will start at ground 0.

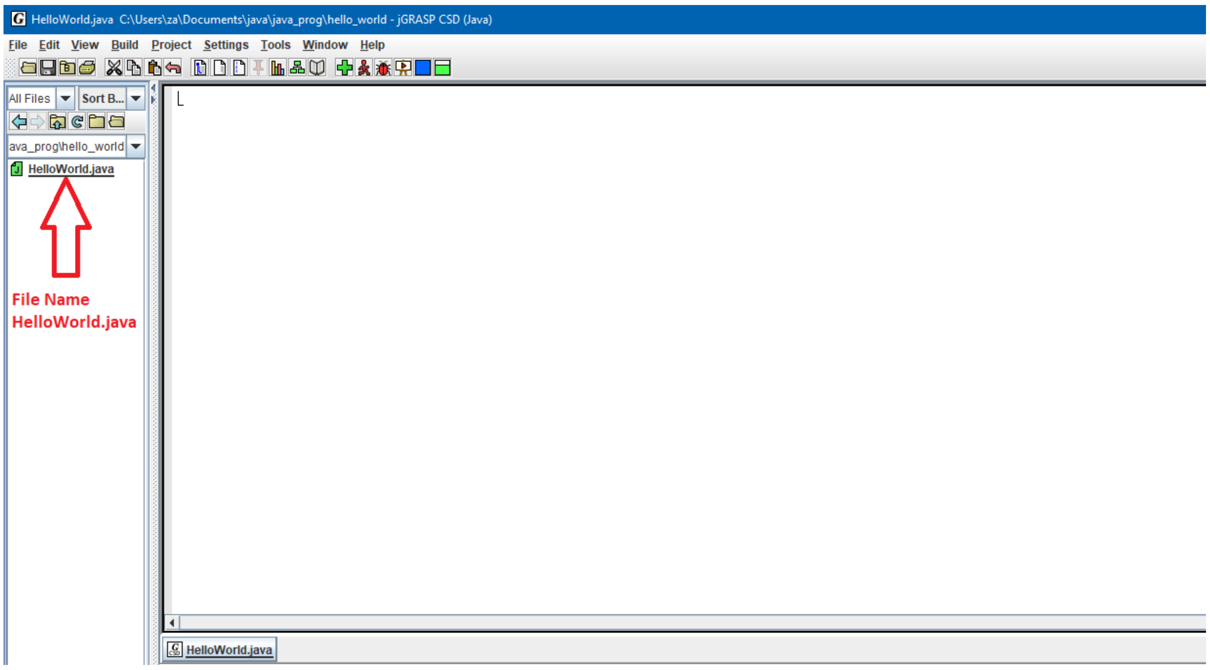
4.1 Creating a New Java File and Saving it

1. Open up JGrasp.

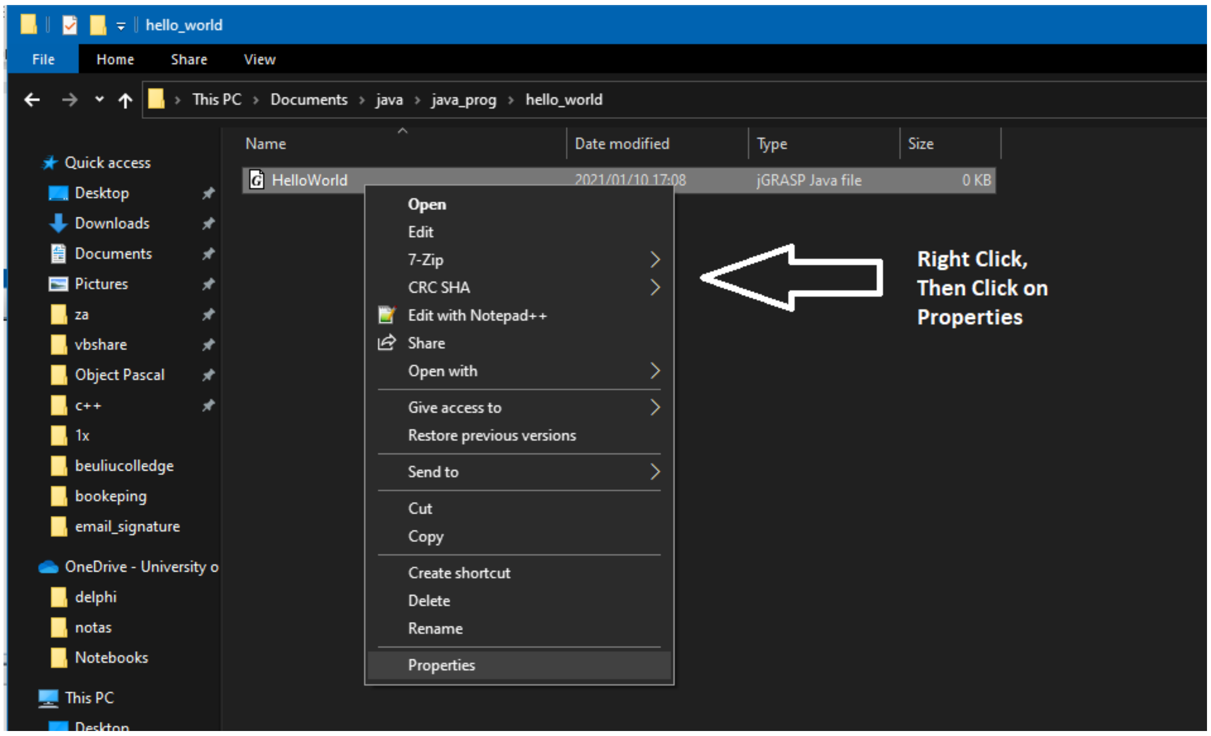
2. Go to File -> New -> Java

3. Go to File -> Save As .

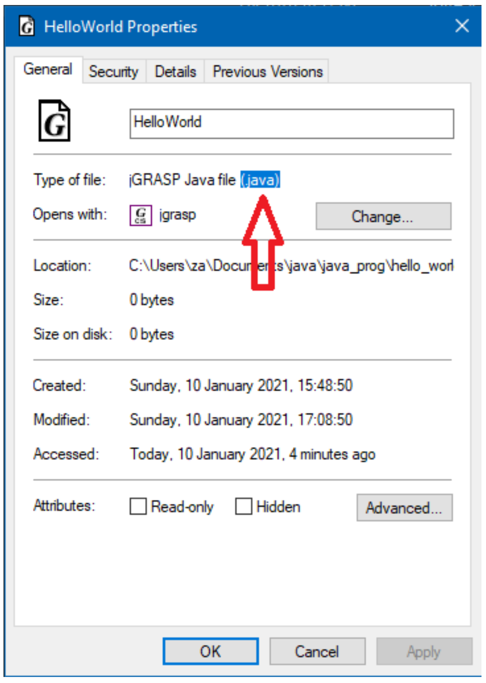
- Save the File as HelloWorld


HelloWorld.java C:\Users\za\Documents\java\java_prog\hello_world - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
All Files
Sort B...
ava_prog\hello_world
HelloWorld.java
File Name
HelloWorld.java
HelloWorld.java

To confirm you file has been saved correctly, check it’s file type



The file Type should be .java


HelloWorld Properties

General | Security | Details | Previous Versions

HelloWorld

Type of file: jGRASP Java file (.java)
Opens with: jgrasp
Location: C:\Users\za\Documents\java\java_prog\hello_worl
Size: 0 bytes
Size on disk: 0 bytes
Created: Sunday, 10 January 2021, 15:48:50
Modified: Sunday, 10 January 2021, 17:08:50
Accessed: Today, 10 January 2021, 4 minutes ago
Attributes: ☐ Read-only ☐ Hidden

4.2 Java Skeleton

Every program you write from now till Grade 12 will have this skeleton. We are not going to go into the details of the skeleton(at this point). For now you should just know that you will be using it in every program. **The name of the file, should be the exact same as what comes after the class keyword.**

Inside of JGrasp, type the following :

```
public class HelloWorld {
 public static void main(String[] args) {
  }
}
```

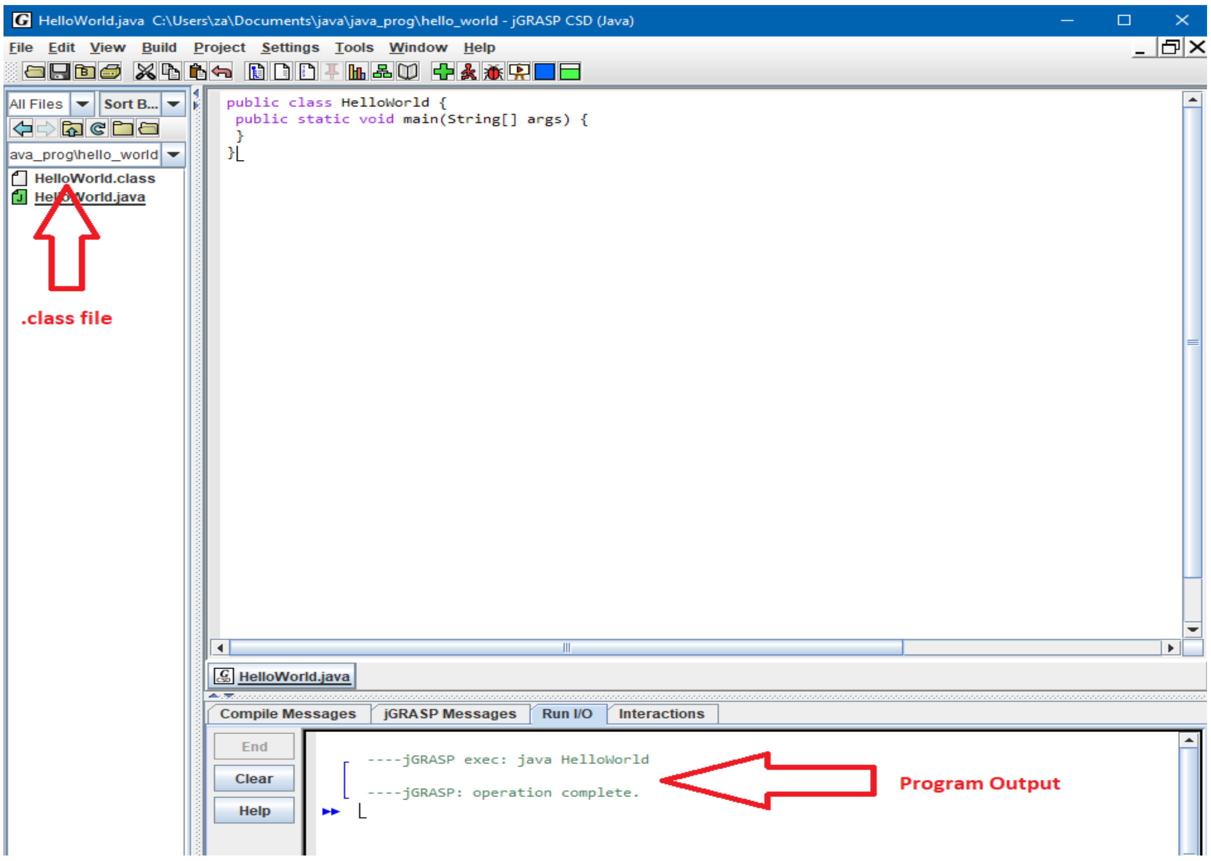
- **Note here the File name and the word after the class keyword are exactly the same.**

Once you have typed the above code inside of jGRASP, we will run the program.

Go to Build -> Run (at the top in the menu bar of jGRASP)

A few things happened here when you clicked the run button.

1. The program was ***compiled*** (the code that we wrote, was translated into machine language which the processor understands) by the Java Compiler
2. A .class file was created. You can check the directory where you HelloWorld.java file is saved.
3. The output was produced (however there is nothing in it as this point)



```
public class HelloWorld {
 public static void main(String[] args) {
  }
}
```

```
----jGRASP exec: java HelloWorld

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

4.2.1 Adding Comments

Comments are ignored by the Java Compiler, meaning that they are not part of the code. A Comment in Java is anything preceding //

We are now going to add two comments to our HelloWorld.java

Update the code so it now looks like this :

```
public class HelloWorld {
 public static void main(String[] args) {
 } // end of HelloWorld Class
} // end of main
```

You can use the shortcuts CTRL+S to save the program, and CTRL + R to run the program to make sure there are no errors.

4.3 OUTPUTTING HELLO WORLD

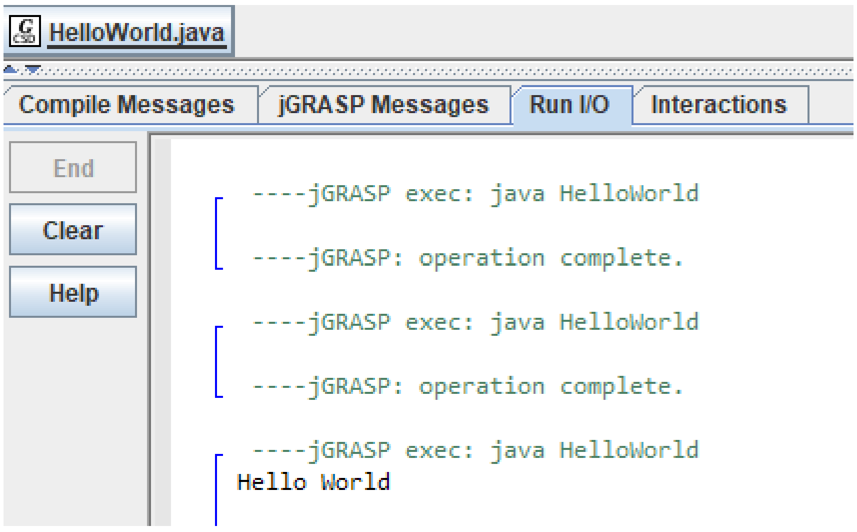
To print something to the console, we use

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

The text inside of the apostrophes is what will be printed.

Update the code to print out Hello World:

```
public class HelloWorld {
 public static void main(String[] args) {
 System.out.println("Hello World");
 } // end of HelloWorld Class
} // end of main
```



```
 ----jGRASP exec: java HelloWorld

 ----jGRASP: operation complete.

 ----jGRASP exec: java HelloWorld

 ----jGRASP: operation complete.

 ----jGRASP exec: java HelloWorld
Hello World
```

5 JAVA DATA TYPES

We are going to look at two definitions regarding a computer :

Input : Data that a computer receives.

Output : Data that the computer sends.

If we tell a computer what is 1 + 1. The 1+1 is the *Input* and the answer 2 is the *Output.*

In Java, there are various **Data Types** that allow us to store various types of Input, that exist in the real world. We will look at three basic data Types below :

String : Strings are all the letters from A -Z, including all characters. The “Hello World” text is considered to be a string. Anything inside of and open and closed apostrophe is treated as a string in Java

Example:

- “Hello World”.
- “123”
- “!@##$#$@#$#@”

Integer : Integers have the same meaning as they do in Mathematics. They are whole numbers(not fractional) that can be negative, positive and zero.

Example :

- 1
- 0
- -1

Floating Point or Just Float : Float data types are Fractional Numbers;

Example:

- 1.23
- -3.321
- 0.00

Later on in this document we will look at more data types.

5.1 Variables

Variables are used for storing data.

To create a variable, use the syntax :

```
type variable = value
```

- Type : Int, string, Float
- Variable : the name of the variable
- Value : the value you want to store inside of the variable

Example:

- We want to store the name Jessie for further use in our program. First of all Jessie is a bunch of letters/characters. Therefore the string data type seems most appropriate.
  - `String name = "Jessie";`
- We want to store the number 123 inside of our program. The Integer data type is most suitable :
  - `int number = 123;`
- We want to store the floating point number 123.321 inside of our program. The floating point data type is most suitable:
  - `float num = 123.321f;`

*take note of the f at the end of floating point numbers

5.2 Extending the Hello World Program

Our HelloWorld Program should look like this at the current moment:

```
public class HelloWorld {
 public static void main(String[] args) {
 System.out.println("Hello World");
 } // end of HelloWorld Class
} // end of main
```

Lets delete the *System.out.println* line, so just the skeleton remains. CTRL+S and CTRL+R to save and run your program.

```
public class HelloWorld {
 public static void main(String[] args) {
 } // end of HelloWorld Class
} // end of main
```

Now declare a String variable called Hello, and store the value *HelloWorld* inside of it.

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 } // end of HelloWorld Class
} // end of main
```

Save and run your program, to make there are no errors.

Now write the code, to Output the String variable Hello:

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 System.out.println(Hello);
 } // end of HelloWorld Class
} // end of main
```

Println : prints to a new line.

To see this in action, **create a new string variable called java, and store the value “Java is great!” inside of it. Then add another println statement to output the variable Java**.

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 String java = "Java is great!";
 System.out.println(Hello);
 System.out.println(java);
 } // end of HelloWorld Class
} // end of main
```

Notice in the output, HelloWorld and Java is great! Are on separate lines.

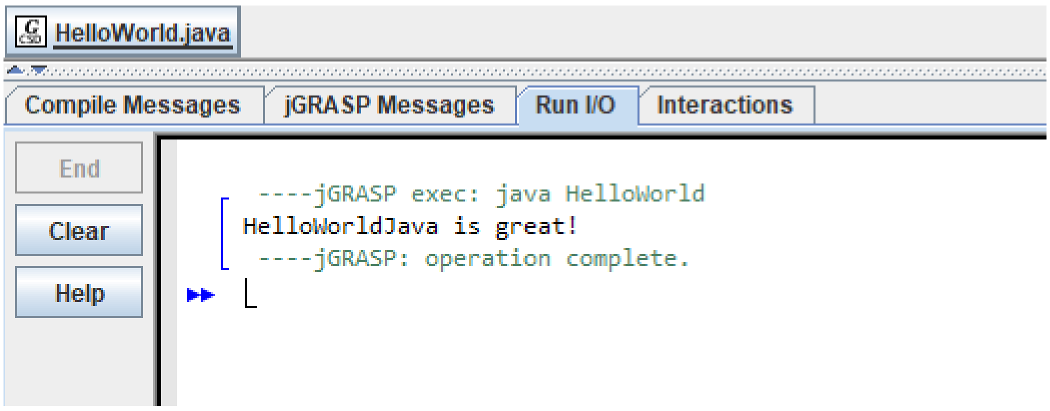
5.2.1.1 Print vs Println

Using the **print** statement, as oppose to the **println** statement will print to the same line.

Remove the two System.out.println statements, and replace them with System.out.print

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 String java = "Java is great!";
 System.out.print(Hello);
 System.out.print(java);
 } // end of HelloWorld Class
} // end of main
```

Notice in the output, that Hello and Java is great! Are now on the same line.



```
 ----jGRASP exec: java HelloWorld
HelloWorldJava is great!
 ----jGRASP: operation complete.
```

5.2.1.2 Concatenating Two Strings

The word **concatenate** means to join two or more strings.

Remove the two System.out.print statements from your code

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 String java = "Java is great!";
 } // end of HelloWorld Class
} // end of main
```

We will now concatenate our two strings variables *Hello* and *java*, using the + operator.

```
public class HelloWorld {
 public static void main(String[] args) {
 String Hello = "HelloWorld";
 String java = "Java is great!";

 System.out.print(Hello + " " + java);

 } // end of HelloWorld Class
} // end of main
```

Look at the output, there is now a space in between the *Hello* and *Java is great!* Statements.

6 DATA TYPES PROGRAM

In this section we are going to be learning to work with various Java Data Types. Before we proceed lets look at the different types of data types that allowed in Java.

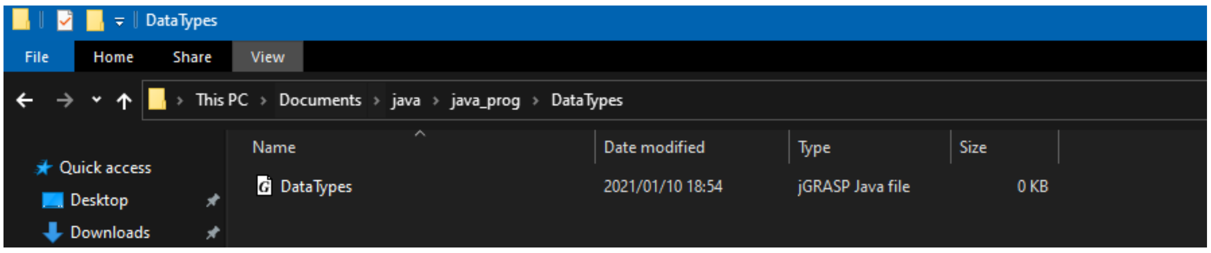
- Boolean : true of false values
- Char : Single letter characters
- Integer : Positive, Negative whole numbers
- Double : Fractional Numbers
- Float : Fractional Numbers

You might be tempted to ask, what is the difference between float vs double. The answer is they way they are stored in computer memory. Float takes up 4 bytes of computer memory, whilst Double takes up 8 bytes of computer memory (meaning larger numbers are allowed to be stored).

Start A New Program in jGRASP. Go To File -> New - >Java

Save the Program as DataTypes.java

- Check the folder, to see if your file is saved correctly.



Name	Date modified	Type	Size
DataTypes	2021/01/10 18:54	jGRASP Java file	0 KB

Inside of jGRASP create the program skeleton :

```
public class DataTypes {
 public static void main(String[] args) {
  }
}
```

CTRL+S to save the program, and CTRL+R to run it, to confirm there are no errors.

Declare a Boolean variable called truth, assign it the value true:

```
boolean truth = true;
```

Declare a Char variable called letter, assign it the value a:

```
char letter = 'a';
```

Declare a Integer variable called number, assign it the value 1:

```
int number = 1;
```

Declare a Decimal variable called fraction, assign it the value 1.23:

```
double fraction = 1.23;
```

Declare a Float variable called fraction1, assign the value 1.233:

```
float fraction1 = 1.23f;
```

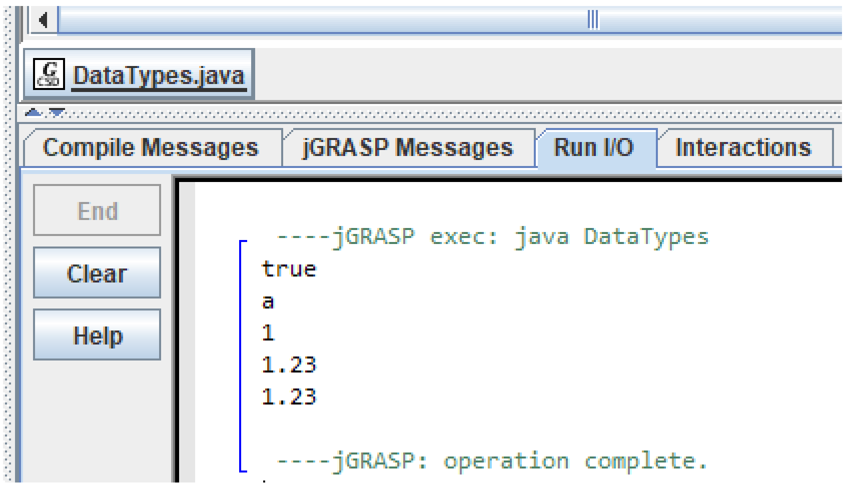
Your program code should now look like this :

```java
public class DataTypes {
 public static void main(String[] args) {
 boolean truth = true;
 char letter = 'a';
 int number = 1;
 double fraction = 1.23;
 float fraction1 = 1.23f;
 }
}
```

We will now add in 5x System.out.println statements to output each of the 5 variables we just created:

```java
public class DataTypes {
 public static void main(String[] args) {
 boolean truth = true;
 char letter = 'a';
 int number = 1;
 double fraction = 1.23;
 float fraction1 = 1.23f;

 System.out.println(truth);
 System.out.println(letter);
 System.out.println(number);
 System.out.println(fraction);
 System.out.println(fraction1);
 }
}
```

Run your program, and the output should look like this :



```
 ----jGRASP exec: java DataTypes
true
a
1
1.23
1.23

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

7 Arithmetic Operators

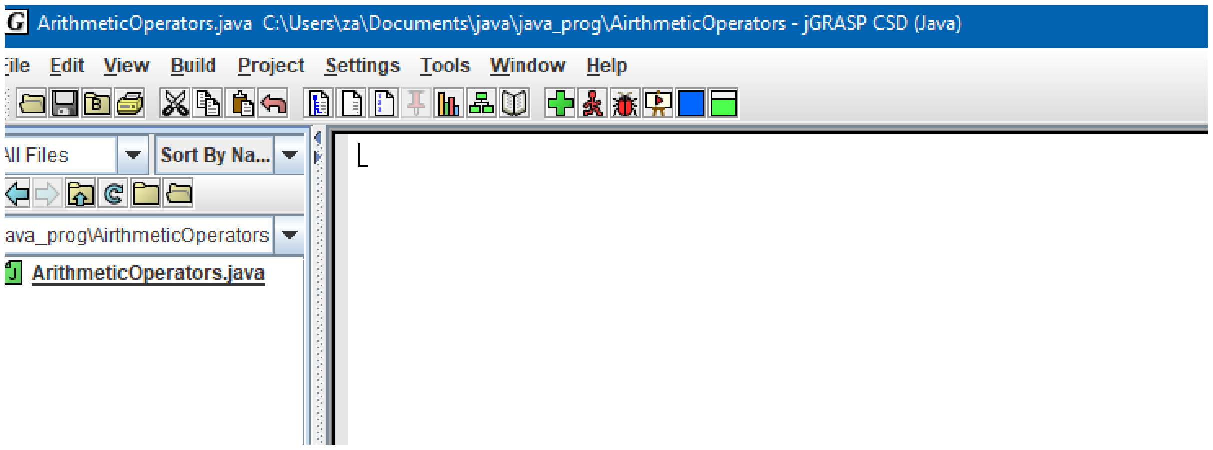
In this section we will be looking at basic mathematic operations that can be performed in Java.

An operator is used to perform operations on variables and values.

Operator	Use	Description	Example
+	x + y	Adds x and y	int num = 1 + 1
-	x – y	Subtracts y from x	Int num = 1 - 1;
*	x * y	Multiplies x by y	int num = 1 * 1
/	x / y	Divides x by y	int num = 1 / 1
%	x % y	Computes the remainder of dividing x by y (modulus)	int num = 1 % 1

Start a new Program. Go to File -> New -> Java

Save the File as ArithmeticOperators.java


ArithmeticOperators.java C:\Users\za\Documents\java\java_prog\AirthmeticOperators - jGRASP CSD (Java)
ile Edit View Build Project Settings Tools Window Help
ll Files
Sort By Na...
ava_prog\AirthmeticOperators
ArithmeticOperators.java

Add in the program skeleton

```
public class ArithmeticOperators{
 public static void main(String[] args){
 }
}
```

Save and Run the program to make sure there are no errors.

Declare two integer variables called x and y, and assign them the values of 2 and 1 respectively:

```
public class ArithmeticOperators{
 public static void main(String[] args){
 int x = 2;
 int y = 1;
 }
}
```

Create a third integer variable called result, assign it a value of 0.

```
int result = 0;
```

Add x to y, and assign the sum to result:

```
sum = x + y;
```

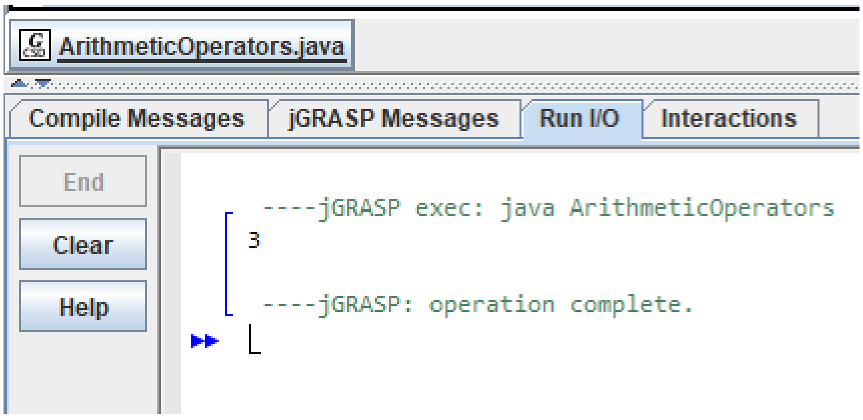
Output the variable result to the console :

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

Your code should now look like this :

```
public class ArithmeticOperators{
 public static void main(String[] args){
 int x = 2;
 int y = 1;
 int result = 0;
 result = x + y;
 System.out.println(result);
 }
}
```

Run your program, and 3 should be outputted to the console.



```
 ----jGRASP exec: java ArithmeticOperators
3

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

Add a new line, that subtracts y from x and stores the result in the result variable.

```
result = x - y;
```

Output the result, by adding in another System.out.println(result);

```
public class ArithmeticOperators{
 public static void main(String[] args){
 int x = 2;
 int y = 1;
 int result = 0;

 result = x + y;
 System.out.println(result);

 result = x - y;
 System.out.println(result);

 }
}
```

Now you might be wondering how did we use the result variable twice ? Remember a Java program runs from top to bottom. At first we assigned result to 0, when we declared it.

```
int result = 0;
```

After this, we assigned it to the value of x + y , where its initial value of 0 was over written by the answer 3.

```
result = x + y;
```

When we used it for the second time, its value was currently 3, and we over wrote it with the value of subtracting y from x, which gave it a value of 1.

```
result = x - y;
```

Add a new line, that multiples x by y, and stores the result inside of the result variable:

```
result = x * y;
```

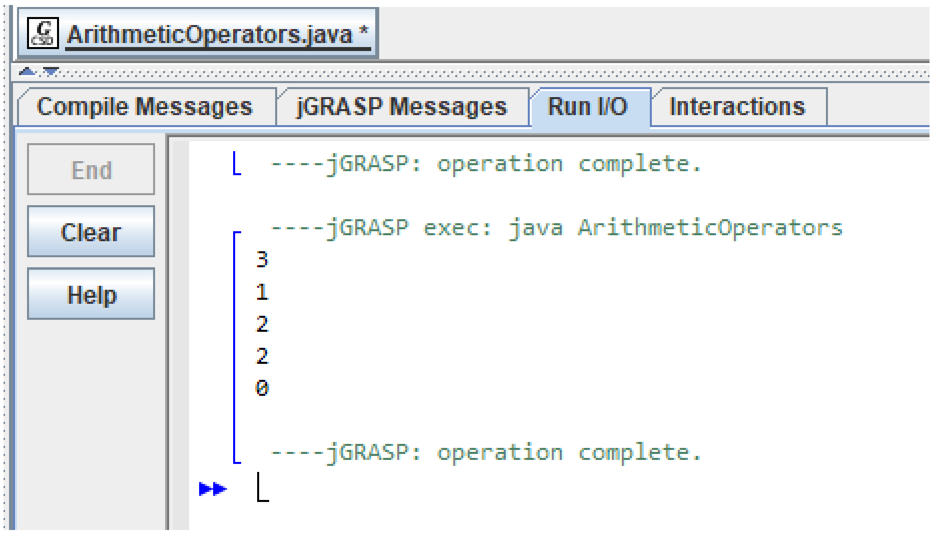
Output the result variable to the console :

```
public class ArithmeticOperators{
 public static void main(String[] args){
 int x = 2;
 int y = 1;
 int result = 0;

 result = x + y;
 System.out.println(result);

 result = x - y;
 System.out.println(result);

 result = x * y;
 System.out.println(result);

 }
}
```

Using this method, you can see how / and % work respectively. The

```
public class ArithmeticOperators{
 public static void main(String[] args){
 int x = 2;
 int y = 1;
 int result = 0;

 result = x + y;
 System.out.println(result);

 result = x - y;
 System.out.println(result);

 result = x * y;
 System.out.println(result);

 result = x / y;
 System.out.println(result);

 result = x % y;
 System.out.println(result);

 }
}
```

You should get the following output :



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

 ----jGRASP exec: java ArithmeticOperators
3
1
2
2
0

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