

Working with variables

We shall now learn how to work with variables in Java and how to manipulate computer memory using variables.

Data is an important part of programming and programs work by operating on data. Data can be text, numbers, pointers, objects or some specified memory location.

Data is accessed by giving it a name and then assigning a value to it. The name of the data and its value is referred to as a **variable**. Now, let's see how we can work with number variables in Java.

To use a number variable in Java, you must specify what type it is i.e. its **data type**. Whole numbers e.g. 5, 13, 28 etc. have a data type of **int** (for integer) while floating point numbers e.g. 1.6, 7.34, 0.8 etc. use a **double** data type. To store a value into a variable we use the equal sign (=) operator.

For example, to store a value **20** to a variable called **age** we write the following:

```
int age;  
age = 15;
```

To workout with a practical example, let's modify inside the "main" part to look like shown below:

```
public static void main(String[] args) {  
    int age;  
    age = 15;  
}
```

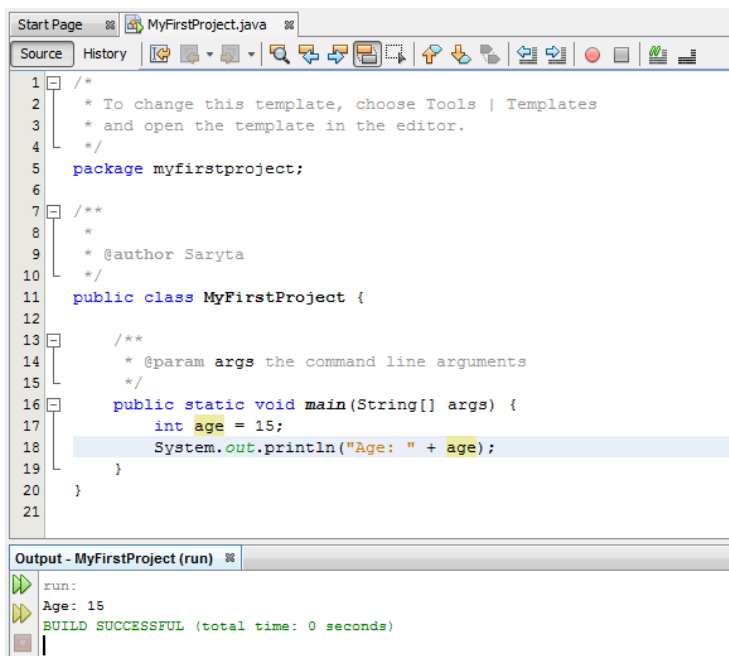
Delete or comment out your Hello World line.

So, to store a whole number, you first type the word **int**, followed by a space and then the name of your integer variable. To assign the value, you type the name of the variable followed by a space, then the equal sign (=) and finally the value itself, don't forget the semicolon.

What the above code mean is that we want to store a value of **20** into an integer variable called "**age**". Notice the above code can also be written using one line like shown below:

```
public static void main(String[] args) {  
    int age = 15;  
}
```

To see the output when you run your program, we have to include the Java function for displaying the output on the output window. So, modify the code again to include the output function **println()**. Your code now should look like shown below:



The screenshot shows an IDE window titled 'MyFirstProject.java'. The code is as follows:

```
1  /*  
2  * To change this template, choose Tools | Templates  
3  * and open the template in the editor.  
4  */  
5  package myfirstproject;  
6  
7  /**  
8  *  
9  * @author Saryta  
10 */  
11 public class MyFirstProject {  
12  
13     /**  
14     * @param args the command line arguments  
15     */  
16     public static void main(String[] args) {  
17         int age = 15;  
18         System.out.println("Age: " + age);  
19     }  
20 }  
21
```

Below the code editor is the 'Output - MyFirstProject (run)' window, which displays the following output:

```
run:  
Age: 15  
BUILD SUCCESSFUL (total time: 0 seconds)
```

What we have between the round brackets of **println** is some direct text enclosed in double quotes and a plus sign, followed by the name of our variable:

```
("Age: " + age );
```

The plus sign tells Java to "join together" the output. So we're joining together the direct text, "Age: ", and the name of our variable, **age**. The joining together is known as **concatenation**. Notice each line of the code ends with a semicolon (;). Now, run the program.

The value of **age** that we stored in the variable called "age" is output after the equals sign.

Variables in Java can be called any name but Java has some rules to be followed when naming your variables. Here are some of the rules:-

- A variable name cannot start with a number. So you cannot call a variable **1stsalary** or **20years**.
- A variable name cannot be the same as a keyword used in Java. Keywords in Java NetBeans appear in blue in the code window, like **int**, so you can't miss them.
- A variable name cannot have a space e.g. my Age. If you want to have two words for a variable name, just join them e.g. myAge or use an underscore e.g. my_age.
- Variable names are case sensitive. So **Age** and **age** are different variable names.

Arithmetic operators in Java

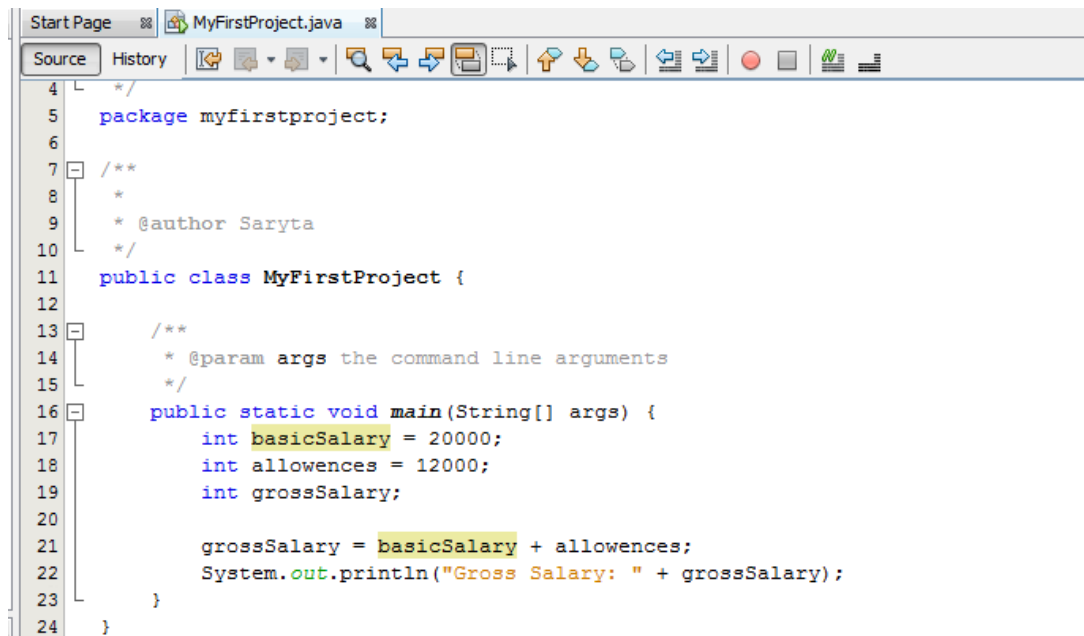
Java uses the commonly known computer arithmetic operators for calculations:

- +
 -
 - *
 - /
- (the plus sign) for addition
(the minus sign) for subtraction
(the asterisk sign) for multiplication and
(the forward slash) for division

Working with variables of int type

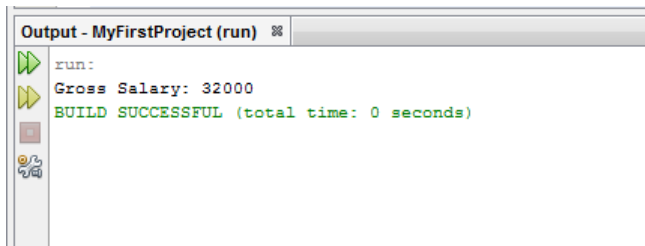
Let's try some simple calculations. Modify your code and add three **int** variables. One to store **basicSalary**, another to store **allowances** and another to store **grossSalary**. Go ahead and assign 20000 to **basicSalary** and 12000 to **allowances**. What we want to do here is to add **basicSalary** and **allowances** and store the sum value in the variable called **grossSalary** and then output the **grossSalary** as the sum value.

Your code now should look like shown below:



```
4  */
5  package myfirstproject;
6
7  /**
8   *
9   * @author Saryta
10  */
11  public class MyFirstProject {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          int basicSalary = 20000;
18          int allowances = 12000;
19          int grossSalary;
20
21          grossSalary = basicSalary + allowances;
22          System.out.println("Gross Salary: " + grossSalary);
23      }
24  }
```

Run the program and you should be able to see the output. You'll see that the values in **basicSalary (20000)** and **allowances (12000)** have been summed up and stored in **grossSalary** as **32000**, then **32000** is displayed together with the text..



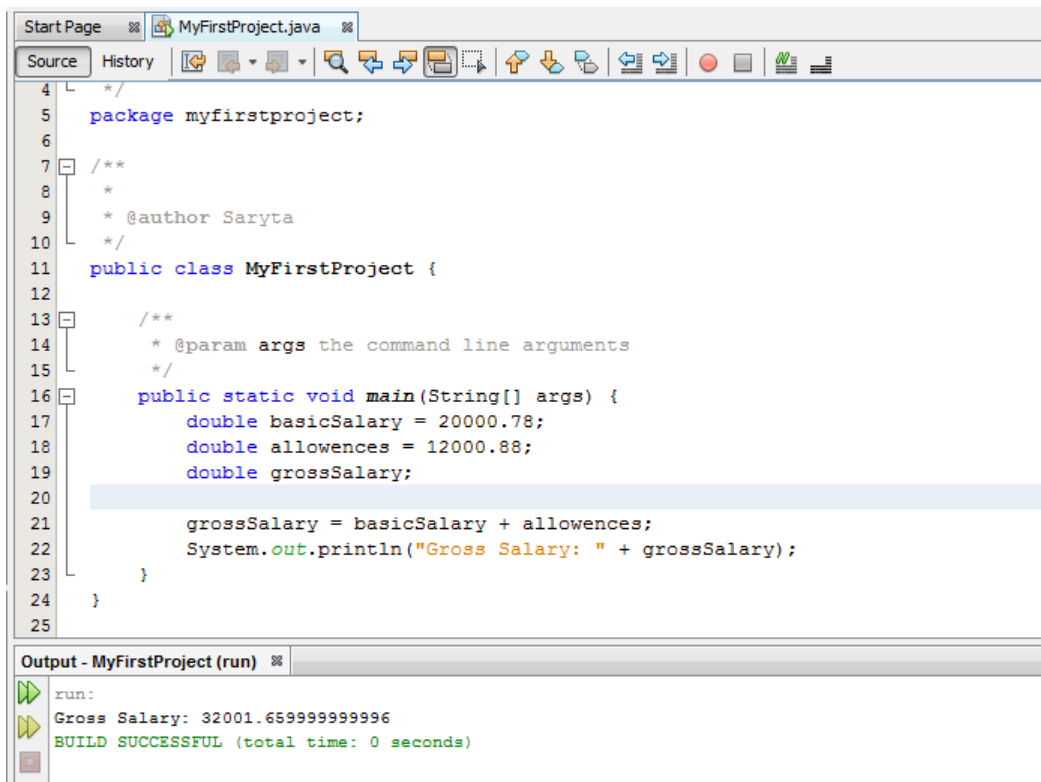
```
Output - MyFirstProject (run) %
run:
Gross Salary: 32000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java can store quite large numbers in the **int** variable type. The maximum value is **2147483647**. If you want a minus number the lowest value you can have is **-2147483648**. If you want larger or smaller numbers than that then you can use a number variable of type **double**.

The **double** variable type is also used to hold **floating** point numbers i.e. numbers with a decimal point like 18.7, 10.3, 154.108 etc. If you store a **floating** point value in an **int** variable, NetBeans will underline the faulty code and if you run the program, the compiler will display an error message.

Working with variables of double type

Up to this point you've been working with variables of **int** type. Let us now try to work with variables of type **double**. Modify your code to make your variables to be of type **double**. Then assign double type values to these variables like shown below. Notice that because we expect a **double** variable as the output, we also need to change the type of our output variable (**grossSalary**) from **int** to **double** type. Run the program again, this time you should be able to get a double output.

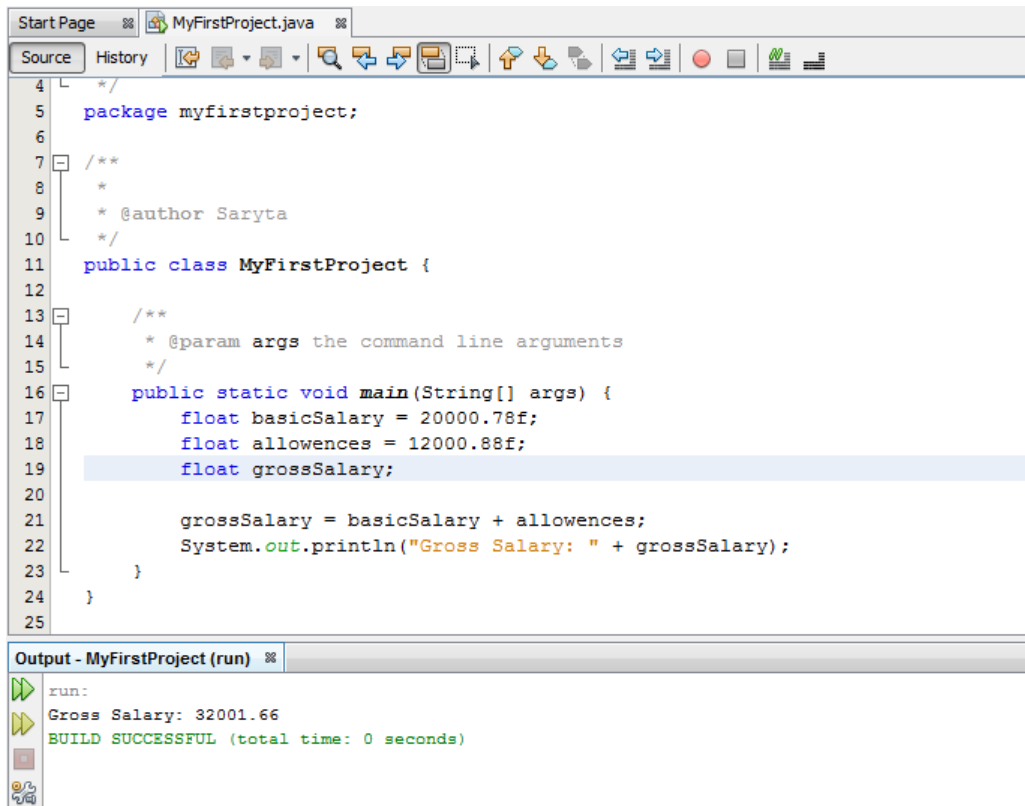


```
Start Page  MyFirstProject.java
Source History
4  /*
5  package myfirstproject;
6
7  /**
8   *
9   * @author Saryta
10  */
11  public class MyFirstProject {
12
13      /**
14       * @param args the command line arguments
15      */
16      public static void main(String[] args) {
17          double basicSalary = 20000.78;
18          double allowances = 12000.88;
19          double grossSalary;
20
21          grossSalary = basicSalary + allowances;
22          System.out.println("Gross Salary: " + grossSalary);
23      }
24  }
25
Output - MyFirstProject (run) %
run:
Gross Salary: 32001.659999999996
BUILD SUCCESSFUL (total time: 0 seconds)
```

Working with variable of float type

The **double** values can store really big numbers of the floating point type. So, Instead of using **double**, **float** type can be used. When storing a value of a **float** type, we add letter "f" at the end of that value but before the semicolon. See below.

To experiment with a **float** data type values, modify your code again like shown below and run it.



```
4  */
5  package myfirstproject;
6
7  /**
8   *
9   * @author Saryta
10  */
11 public class MyFirstProject {
12
13     /**
14      * @param args the command line arguments
15      */
16     public static void main(String[] args) {
17         float basicSalary = 20000.78f;
18         float allowances = 12000.88f;
19         float grossSalary;
20
21         grossSalary = basicSalary + allowances;
22         System.out.println("Gross Salary: " + grossSalary);
23     }
24 }
25
```

Output - MyFirstProject (run)

```
run:
Gross Salary: 32001.66
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java order of Operators Precedence

The order in which arithmetic's operations are executed in Java is very important to understand. This order is called **Operator Precedence**.

- Multiplication and Division are treated equally, but have high priority than Addition and Subtraction
- Addition and Subtraction are treated equally but have a lower priority than multiplication and division

If you are going to use more than one arithmetic operator in one statement, it a good practice to always use some brackets so as to avoid some output errors derived from operators' precedence.

Working with variables of String type (text)

As well as storing number values, variables can also store text values. Some text type variables can only store one character while others can store lots of characters. To store just one character, the **char** type variable is used, but to store more than one character **String** type variable is used. Mostly you will use **String** type variable as you may need to store lots of characters. We'll create two **String** variables to hold the name of a person.

To define a string variable, type the word **String** followed by a name for your variable. Note that there's an uppercase "S" for **String** and the line ends with a semicolon.

```
String sur_name;
String first_name;
```

To assign values to the string variables, type an equal sign and then the text which should be between two sets of double quotes:

```
sur_name = "Schaerer";
first_name = "Saryta";
```

Remember the above code can also be written as:

```
String sur_name = "Schaerer";  
String first_name = "Saryta";
```

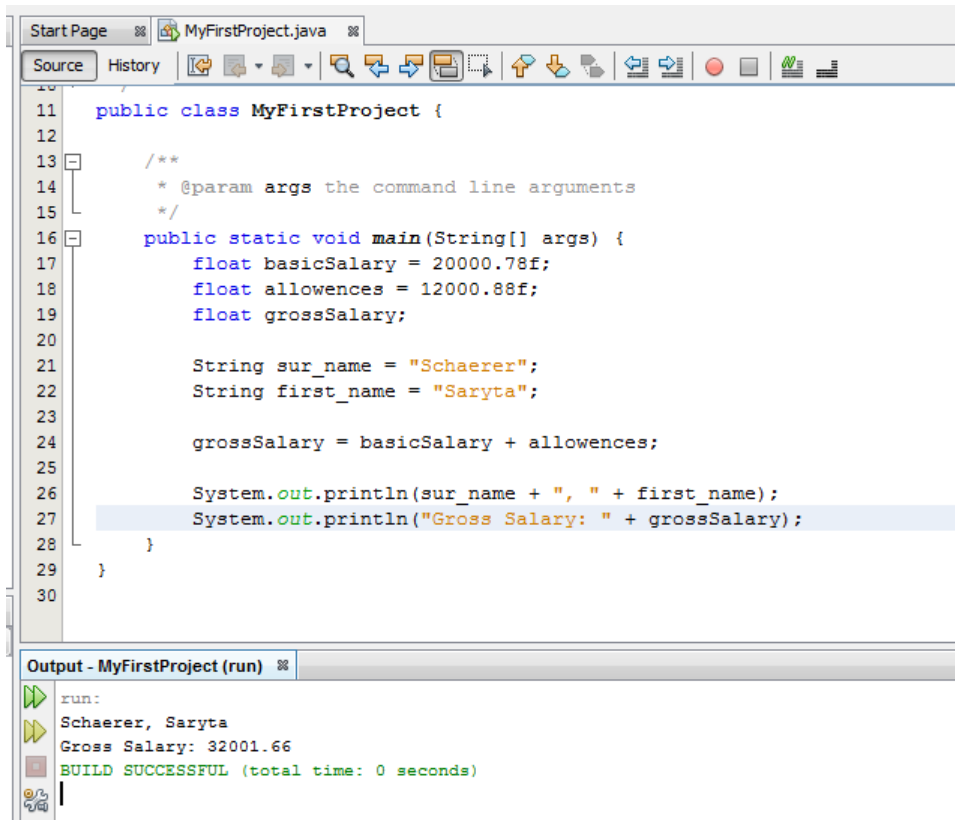
To display both names, add the following code:

```
System.out.println(sur_name + " " + first_name);
```

In between the round brackets of println(), we have this:

```
sur_name + " " + first_name
```

We are telling Java to display the two names together but with a space in between. Now, modify your code like shown below and run once again.



The screenshot shows an IDE window titled 'MyFirstProject.java'. The code is as follows:

```
11 public class MyFirstProject {  
12  
13     /**  
14      * @param args the command line arguments  
15      */  
16     public static void main(String[] args) {  
17         float basicSalary = 20000.78f;  
18         float allowances = 12000.88f;  
19         float grossSalary;  
20  
21         String sur_name = "Schaerer";  
22         String first_name = "Saryta";  
23  
24         grossSalary = basicSalary + allowances;  
25  
26         System.out.println(sur_name + " " + first_name);  
27         System.out.println("Gross Salary: " + grossSalary);  
28     }  
29 }  
30
```

The output window at the bottom shows the following text:

```
run:  
Schaerer, Saryta  
Gross Salary: 32001.66  
BUILD SUCCESSFUL (total time: 0 seconds)
```

If you want to store a single character, use a variable of type **char** (lowercase "c") and then surround the character with single quotes and not double quotes:

```
char grade = 'B';
```

Remember: A String variable has double quotes and a char variable has single quotes.