Lecture (02)
Variables and Arithmetic Operations

Dr. Ahmed ElShafee

Agenda

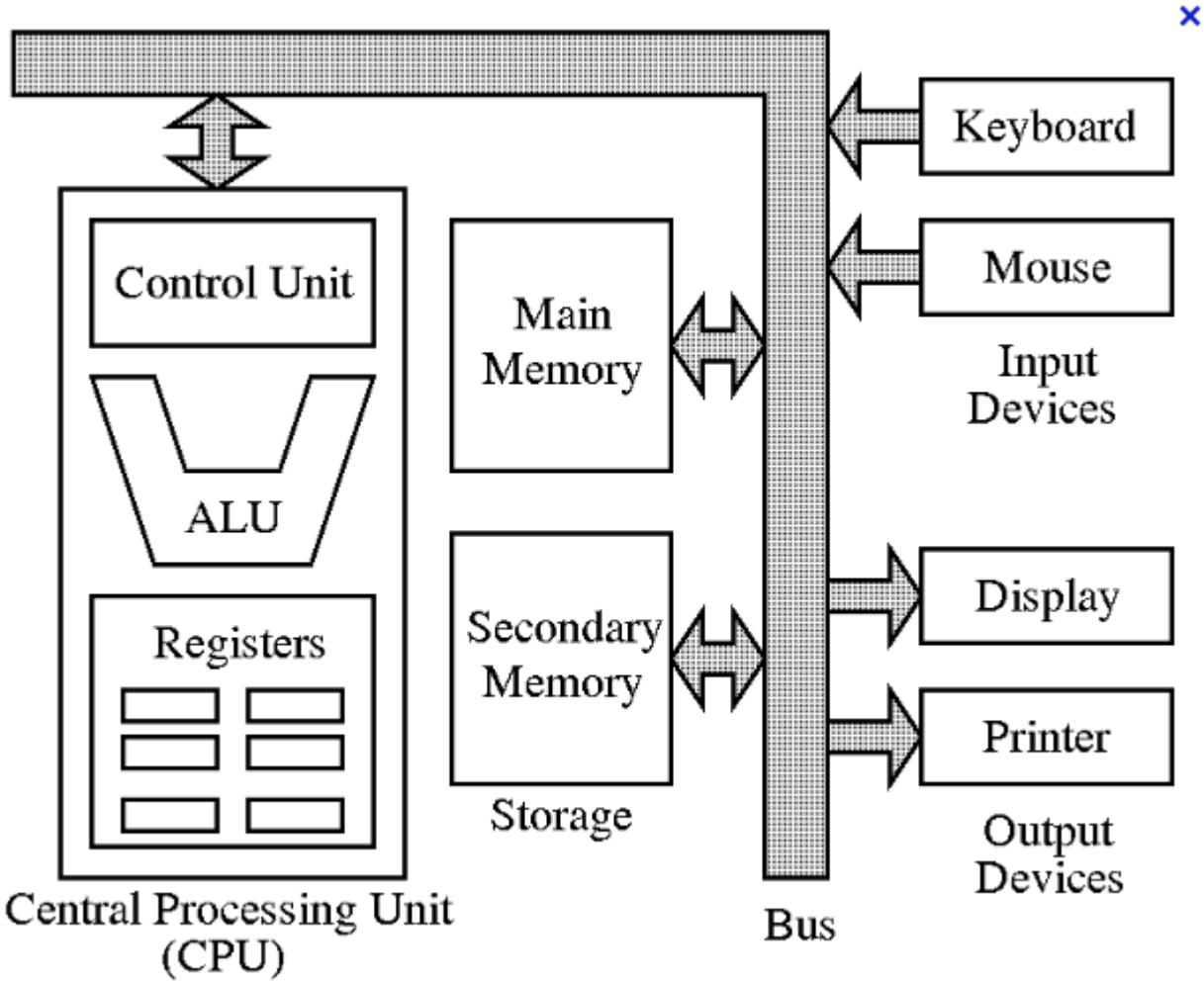
- Variables
- The need for the variables
- Integers
- The need for data types
- The Double variable
- Short and Float
- Simple arithmetic
- Operator Precedence

Variables

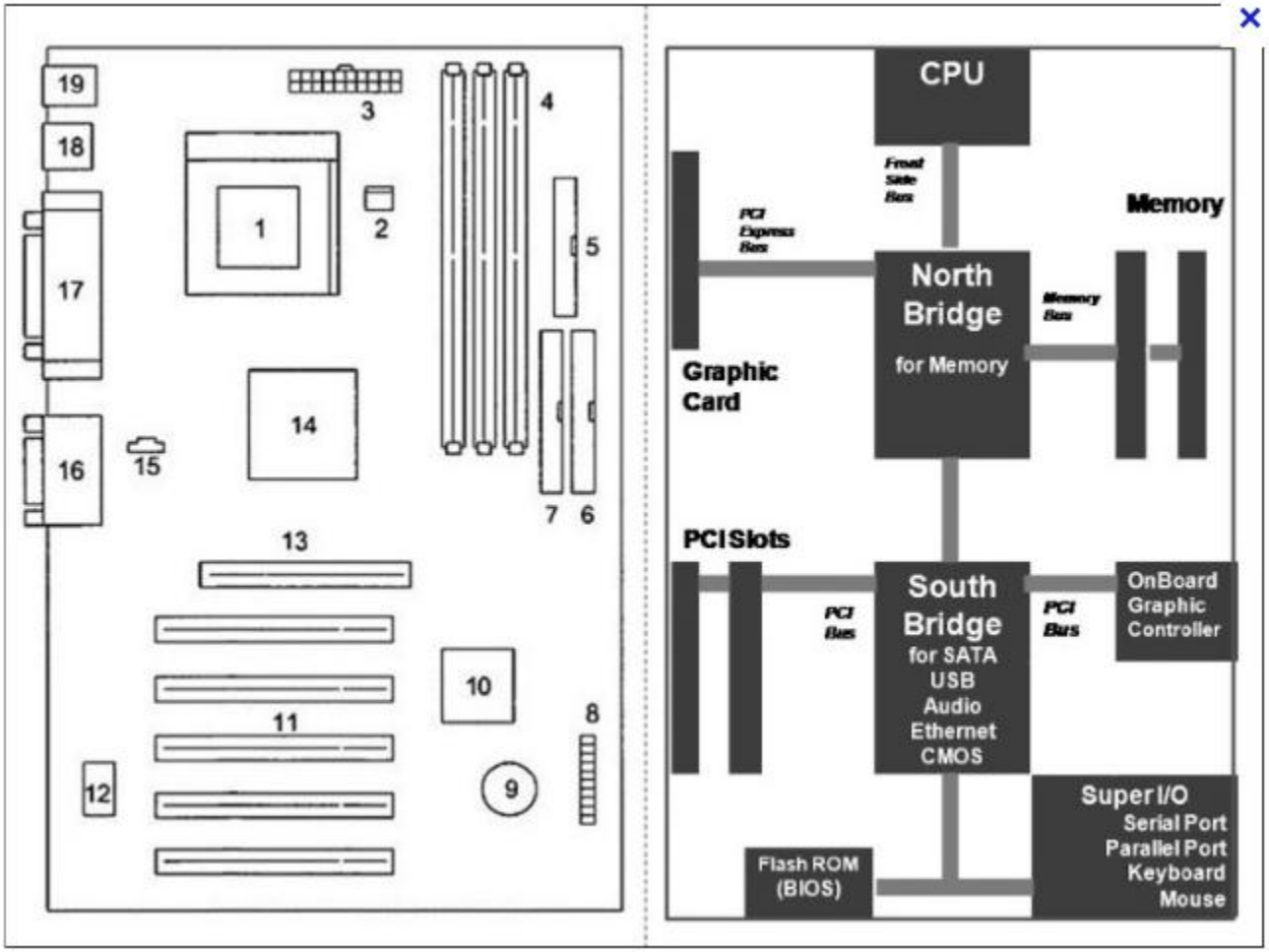
- Programs work by manipulating data placed in memory.
- The data can be numbers, text, objects, pointers to other memory areas, and more besides.
- The data is given a name, so that it can be re-called whenever it is need.
- The name, and its value, is known as a Variable.

The need for variables?

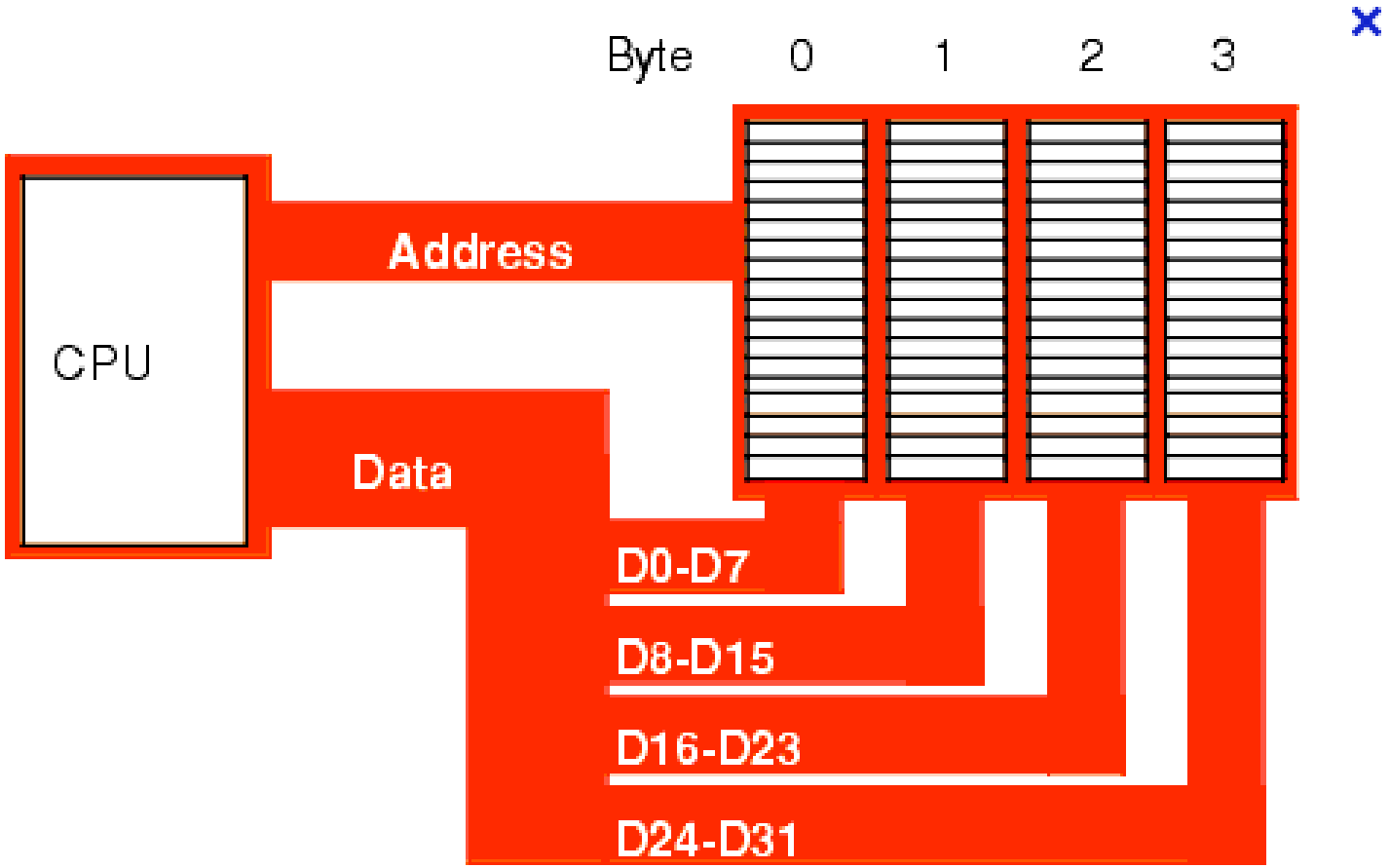
- Lets get back to computer architecture



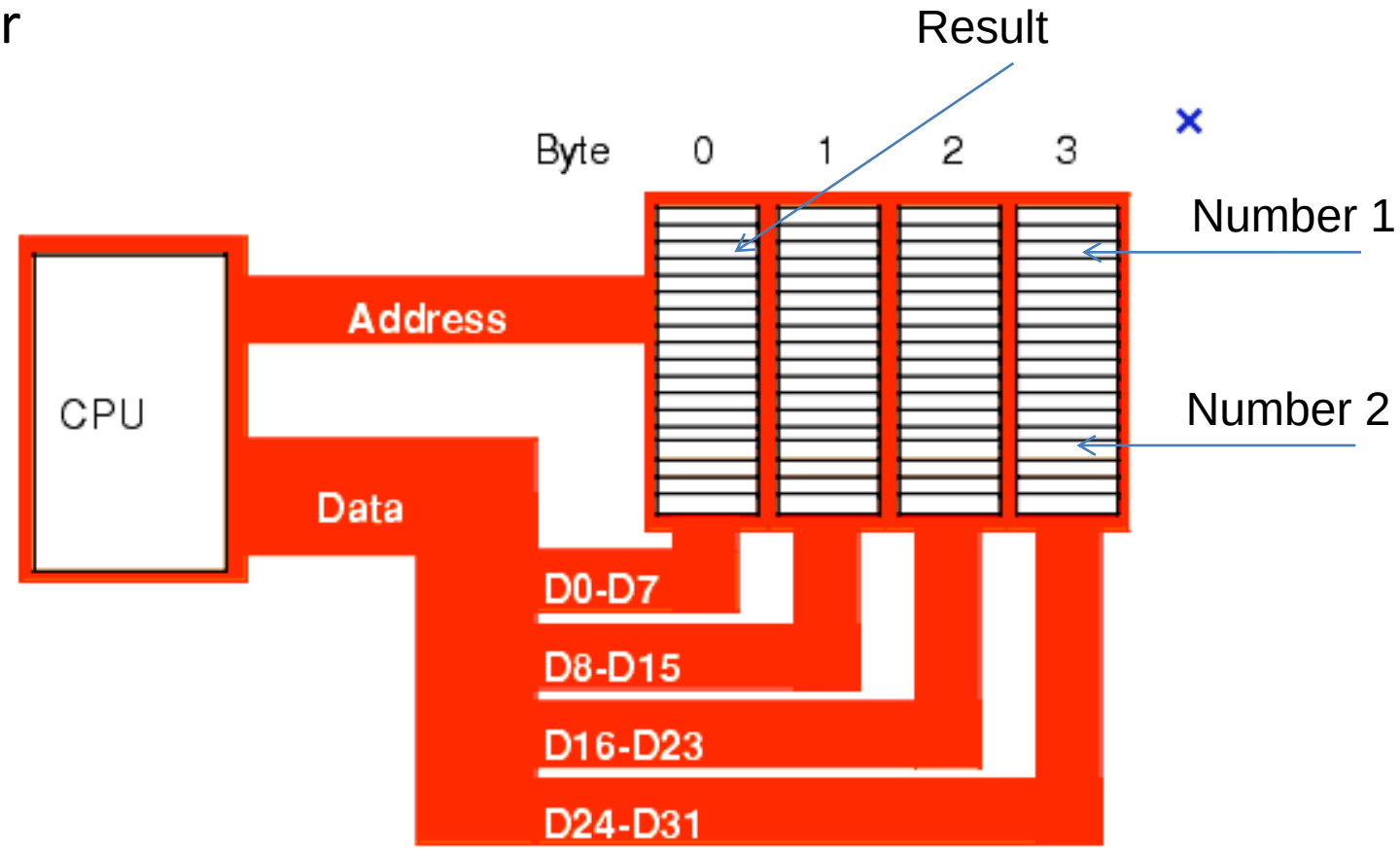
- Focus now on computer ports



- Now focus on the memory it self



- The idea of using variables in to use any available space in memory, and refer to it by unique name assigned the program developer



Integers

- So to tell Java that you want to store a whole number, you first type the word **int,** followed by a space (32 bits).
- You then need to come up with a name for your integer variable.
- So to set up a whole number (integer),

int first_number;

1. Variable names can't start with a number.
   - So first_number is OK, but not 1st_number.
   - You can have numbers elsewhere in the variable name, just not at the start.
2. Variable names can't be the same as Java keywords.
3. There are quite a lot of these, and they will turn blue in NetBeans, like **int** above**.**
4. You can't have spaces in your variable names.

- The variable declaration

int first number

will get you an error.

- **We've used the underscore character,** but it's common practise to have the first word start with a lowercase letter and the second or subsequent words in uppercase:

firstNumber,

myFirstNumber

5. Variable names are case sensitive.
   - So firstNumber and FirstNumber are different variable names.

- To store something in the variable called first_number, you type an equals sign and then the value you want to store

int first_number;

first_number = 10;

- If you prefer, you can do all this on one line

int first_number = 10;

- To print the integer value saved in a certain variable use concatenation sign, type variable name inside println statement, or use concatenation sign to print value after a string.

System.out.println(first_number);

System.out.println("First number = " + first_number);

FirstProject02

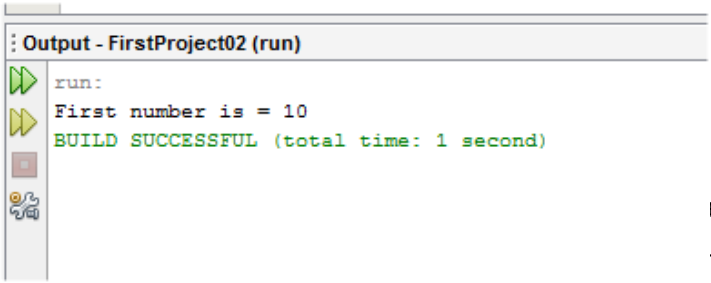
FirstProject02 example;

```
/*...*/
package firstproject02;

/**...*/
public class FirstProject02 {

    /**...*/
    public static void main(String[] args) {
        int first_number;
        first_number=10;
        System.out.println("First number is = "+first_number);
    }
}
```

- Run your program and you should see the following in the Output window at the bottom:



- Let’s try some simple addition.
- Add two more int variables to your code, one to store a second number, and one to store an answer:

int first_number, second_number, answer;

- Notice how we have three variable names on the same line.
- You can do this in Java, if the variables are of the same type (the **int type, for us).**
- Each variable name is then separated by a comma.
- We can then store something in the new variables:

first_number = 10;

second_number = 20;

answer = first_number + second_number;

FirstProject03

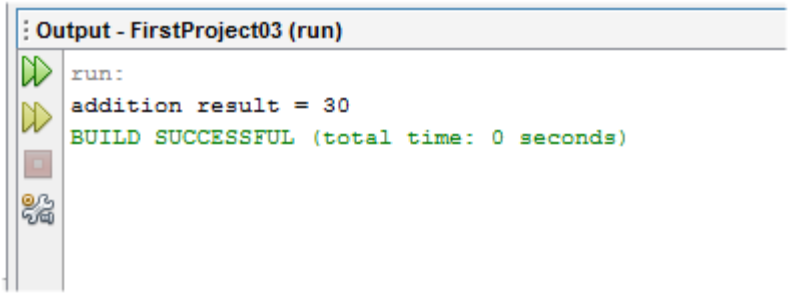
FirstProject03 example:

```
/*...*/
package firstproject03;

/**...*/
public class FirstProject03 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, answer;
        first_number = 10;
        second_number = 20;
        answer = first_number + second_number;
        System.out.println("addition result = "+answer);
    }
}
```

- When you run your program you should get the following in the output window:



- You can also use numbers directly.
- Change the **answer line to this:**

  answer = first_number + second_number + 12;
- Create FirstProject04 example, to find the answer.
- You can store quite large numbers in the **int variable type. The maximum value is** 2147483647 (32 bits).

  (0111 1111 1111 1111 1111 1111 1111 1111)
- If you want a minus number the lowest value you can have is

  -2147483648

  (1000 0000 0000 0000 0000 0000 0000 0000)

FirstProject04

```
/*...*/
package firstproject04;

/**...*/
public class FirstProject04 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, answer;
        first_number = 10;
        second_number = 20;
        answer = first_number + second_number+12;
        System.out.println("addition result = "+answer);
    }
}
```

```
run:
addition result = 42
BUILD SUCCESSFUL (total time: 0 seconds)
```

The need for data types

Memory length	Maximum number
8 bits	255
16 bits	65535
32bits	4294967295

The Double variable

- The double variable (64 Bits) is used to hold floating point values.
- A floating point value is one like 8.7, 12.5, 10.1. In other words, it has a “point something” at the end.
- If you try to store a floating point value in an **int variable, NetBeans will** underline the faulty code.
- If you try to run the program, the compiler will throw up an error message.

FirstProject05

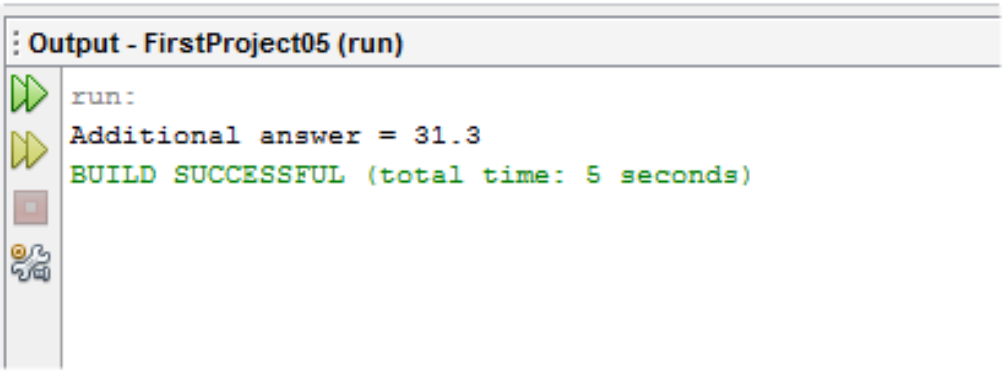
```
/*...*/
package firstproject05;

/**...*/
public class FirstProject05 {

    /**...*/
    public static void main(String[] args) {
        double first_number, second_number, answer;
        first_number = 10.5;
        second_number = 20.8;
        answer = first_number+second_number;
        System.out.println("Additional answer = " + answer);

    }
}
```

Run your program again. The output window should look like this:



Short and Float

- Short (16BITS) is used to store smaller number, and its range is between - 32,768 (1000 0000 0000 0000)

  and +32,767 (0111 1111 1111 1111).
- Instead of using double, float can be used (32 bits).
- When storing a value in a float variable, you need the letter “f” at the end. Like this:

float first_number, second_number, answer;

first_number = 10.5f;

second_number = 20.8f;

Simple arithmetic

- With the variables you've been using, you can use the following symbols to do
- calculations:
- + (the plus sign)
- - (the minus sign)
- * (the multiplication sign is the asterisk)
- / (the divide sign is the forward slash)

FirstProject06

```
/*...*/
package firstproject06;

/**...*/
public class FirstProject06 {

    /**...*/
    public static void main(String[] args) {
        float first_number, second_number, answer;
        first_number = 10.5f;
        second_number = 20.8f;
        answer = first_number / second_number;
        System.out.println("Additional answer = " + answer);

    }
}
```

- When you run the above code, the answer is now 0.5048077.

```
Output - FirstProject06 (run)
run:
Additional answer = 0.5048077
BUILD SUCCESSFUL (total time: 0 seconds)
```

Operator Precedence

- You can, of course, calculate using more than two numbers. But you need to take care of what exactly is being calculated. Take the following as an example:

first_number = 100;

second_number = 75;

third_number = 25;

answer = first_number – second_number + third_number;

- If you did the calculation left to right it would be 100 – 75, which is 25. Then add the third number, which is 25. The total would be 50. However, what if you didn't mean that?

- What if you wanted to add the second and third numbers together, and then deduct the total from the first number? So 75 + 25, which is 100. Then deduct that from the first number, which is 100. The total would now be 0.
- To ensure that Java is doing what you want, you can use round brackets. So the first calculation would be:

answer = (first_number – second_number) + third_number;

FirstProject07

```
1  /*...*/
5  package firstproject07;
6
7  /**...*/
11 public class FirstProject07 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int first_number, second_number, third_number,answer;
18         first_number = 100;
19         second_number = 75;
20         third_number = 25;
21         answer = (first_number - second_number)+third_number;
22         System.out.println("Additional answer = " + answer);
23     }
24 }
25
```

- Answer = 50

```
Output - FirstProject07 (run)
run:
Additional answer = 50
BUILD SUCCESSFUL (total time: 0 seconds)
```

FirstProject08

```
/*...*/
package firstproject08;

/**...*/
public class FirstProject08 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number,answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        answer = first_number - (second_number+third_number);
        System.out.println("Additional answer = " + answer);
    }
}
```

- Answer = 0

```
Output - FirstProject08 (run)
run:
Additional answer = 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

FirstProject09

- Change your math symbols (called Operators) to plus and multiply:

  answer = first_number + second_number * third_number;
- With no brackets, you'd think Java would calculate from left to right.
- So you'd think it would add the first number to the second number to get 175.
- Then you'd think it would multiply by the third number, which is 25.
- So the answer would be 4375.
- Run the program, though. The answer that you actually get is 1975! So what's going on?

```java
/*...*/
package firstproject09;

/**...*/
public class FirstProject09 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number,answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        answer = first_number + second_number*third_number;
        System.out.println("answer = " + answer);
    }
}
```

Output - FirstProject09 (run)

```
run:
answer = 1975
BUILD SUCCESSFUL (total time: 0 seconds)
```

- The reason Java got the “wrong” answer was because of Operator Precedence
- Java treats some mathematical symbols as more important than others.
- It sees multiplication as having a priority over addition, so it does this first.
- It then does the addition.
- So Java is doing this:

answer = first_number + (second_number * third_number);

- With the round brackets in place, you can see that second number is being multiplied by third number.
- The total is then added to the first number.
- So 75 multiplied by 25 is 1875. Add 100 and you get 1975.

FirstProject10

- If you want it the other way round, don't forget to "tell" Java by using round brackets:

answer = (first_number + second_number) * third_number;

```java
/*...*/
package firstproject10;

/**...*/
public class FirstProject10 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number,answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        answer = (first_number + second_number)*third_number;
        System.out.println("answer = " + answer);
    }
}
```

```
Output - FirstProject10 (run)
run:
answer = 4375
BUILD SUCCESSFUL (total time: 0 seconds)
```

FirstProject11

- Division is similar to multiplication: Java does the dividing first, then the addition or subtraction. Change your answer line to the following:

FirstProject11

- **answer = first_number + second_number / third_number;**
- The answer you get is 103.
- Now add some round brackets:

```
/*...*/
package firstproject11;

/**...*/
public class FirstProject11 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number,answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        answer = first_number + second_number/third_number;
        System.out.println("answer = " + answer);
    }
}
```

Output - FirstProject11 (run)

```
run:
answer = 103
BUILD SUCCESSFUL (total time: 0 seconds)
```

FirstProject12

- **answer = (first_number + second_number) / third_number;**
- The answer this time will be 7.
- So without the round brackets, Java does the dividing first, and then adds 100 to the total – it doesn't work from left to right.

```
/*...*/
package firstproject12;

/**...*/
public class FirstProject12 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number,answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        answer = (first_number + second_number)/third_number;
        System.out.println("answer = " + answer);
    }
}
```

```
Output - FirstProject12 (run)
run:
answer = 7
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Here’s a list on Operator Precedence
  - Multiply and Divide – Treated equally, but have priority over Additionand Subtraction
  - Add and Subtract – Treated equally but have a lower priority thanmultiplication and division
- So if you think Java is giving you the wrong answer, remember that Operator Precedence is important, and add some round brackets.

Project 13

- Write a program that execute the following arithmetic expression

1. add 100, to 75
2. multiply result with 25
3. print the whole operation and the final result

Project13

```
/*...*/
package project13;

/**...*/
public class Project13 {

    /**...*/
    public static void main(String[] args) {
        int first__int_number, second_int_number, third_int_number,int_answer;
        first__int_number = 100;
        second_int_number = 75;
        third_int_number = 25;
        int_answer = (first__int_number + second_int_number)*third_int_number;
        System.out.println("("+first__int_number+"+"+second_int_number+")*"+third_int_number+"="+int_answer);
    }
}
```

```
Output - Project13 (run)
run:
100+75*25=4375
BUILD SUCCESSFUL (total time: 0 seconds)
```

Project 14

- Write a program that execute the following arithmetic expression

1. add 100, to 75
2. Divide result by 25
3. print the whole operation and the final result

Project14

```
/*...*/
package project14;

/**...*/
public class Project14 {

    /**...*/
    public static void main(String[] args) {
        int first_number, second_number, third_number;
        float float_answer;
        first_number = 100;
        second_number = 75;
        third_number = 25;
        float_answer = (first_number + second_number)/third_number;
        System.out.println("("+first_number+"+"+second_number+")/"+third_number+"="+float_answer);
    }
}
```

```
Output - Project14 (run)
run:
(100+75)/25=7.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Assignments

1. Create a 3 programs that print the following shapes

2. Create a program that calculate and print the following

(58 + 62)- (22 + 16) * (11 + 5) / 2

(128 / 2) + (39 / 3) * (2 + 14)

((112 * 2) + 5 +(6 / 2)) /2

Thanks,

See you next Lecture, isA