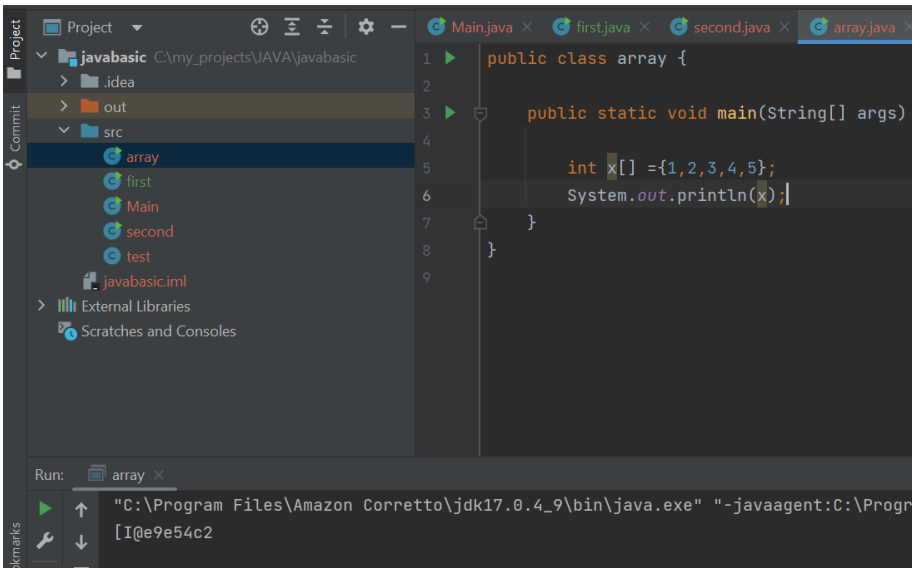
Summary JAVA

Sessions No 5(29-10-2022)

- The **array** is one of the data structures that stores similar data types of values in a continuous manner in RAM. Here in the below example, we have created an **array** "x" which stores five integer values in a continuous manner inside RAM. But when we print the "x" with help of the **System.out** function, we get some physical address. Hence here our array variable "**x**" is storing the address of the first element of the array.
  As we know in JAVA we can't interact with the hardware directly, we have JVM in between so here the address shown in output is not the real physical address of RAM, It is the virtual address provided by JVM, but in reality, JVM knows the real physical address of the first element of the array. Hence, JVM manages pointers in JAVA, but we as developers can't manage pointers because we don't know the real physical address of our variables.



- In the above example datatype of the array tell us which type of data will be stored in the array. Here the data type of the array is "**int**", Then it means we will be storing only "int" values in our array.
-

```
int x[] ={1,2,3,4,5};
System.out.println(x[2]);
```

The Output will be “3”. In most languages including JAVA, the first index number of an array starts from “**0**”.

- Whenever a program is run, It becomes a process in RAM, Now in a process, we have mainly 3 parts **Code Section**, **Stack memory**, And **Heap memory**, Now *Code section* stores the code/instructions we have created in our java code file. *Stack* and *Heap memory* provides space for our variables in RAM.

- Primitive datatypes take space normally from the *Stack memory*(**Primitive data types** include **byte, short, int, long, float, double, boolean, and char**), All the Primitive datatypes take space from stack memory for example -
  Int x =10;
  Here x will get 4 bytes of space from *stack memory* Because here we know the size of “x” is not gonna change(*its value can change but its size is static*)

- **Dynamic arrays** take space from **heap memory**, *Heap memory* gives us functionality that while runtime also we can get more space, Now arrays whose size normally changed while runtime(when more data will be added or some data will be removed) gets the space from heap memory.

- In JAVA, JVM manages heap memory and stack memory for us, IN CPP, Developers have to manage them.

- ```
  int a[]=new int[10];
  ```
  Here we have declared our array “**a**” which can store 10 integer values, now this array will take space from heap memory, size of this array would be 10*4= 40 bytes.

- **Algorithm** - A sequence of statements that collectively achieves a goal

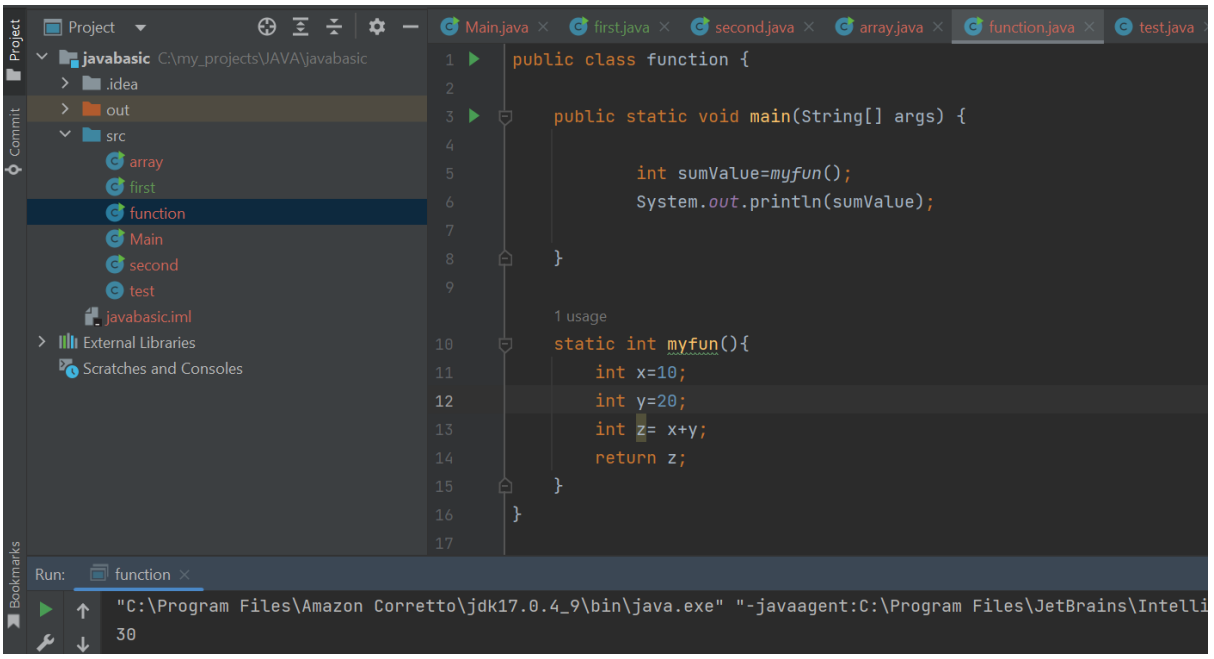
- **Function/Method** - A method is a block of code that only runs when it is called. Now, this block of code we can call at any number of times according to our use. Functions are a good way to manage our code, For each different functionality that we think we will use multiple times, we can create a function for that.

- Create a function in java -
  ```
  static int myfun(){
     int x=10;
     int y=20;
     int z= x+y;
     return z;
  }
  ```
  Here we have created a function named “*myfun*” which adds two values and return the sum of them. Now here as we are returning “**z**”(an int value) that’s why while defining the function we have mentioned its return data type.
  Now we can call this function from our main() function(entry point) and we can store the return value in a different variable.
  ```
  public static void main(String[] args) {
  ```
  ```

```
        int sumValue=myfun();
        System.out.println(sumValue);
```

The output will be - 30

Complete Code -



- JAVA will always call the **main()** function because *main* function is the entry point of our complete code.