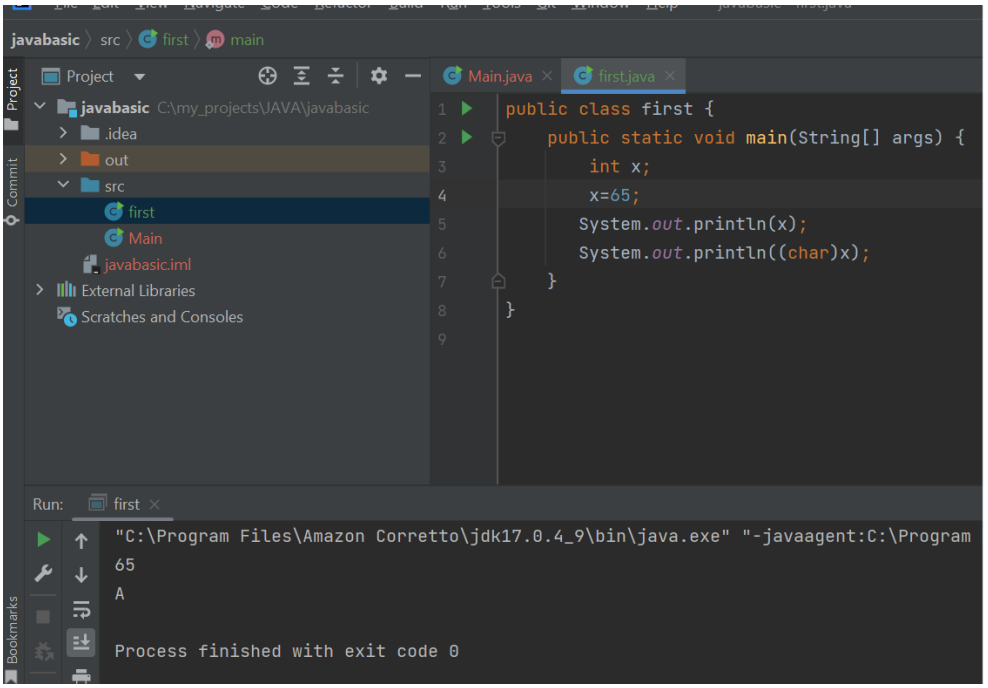
Summary JAVA

Sessions No 3(15-10-2022)

- **Typecasting** is a method of converting one data type to another.

  If I create an *int data type* and store an integer value in it, and if I do “char” typecasting here, Then while reading this value, the CPU will read it as “char”, So in output, you will get the character “A” instead of 65 (Binary notation of A and 65 is same). Check below code.



- In JAVA **System** is the class with which Java Programs interact with the system devices. We will be using this class to interact with standard input and output devices i.e keyboard and console respectively.
- In JAVA we can’t use a variable without assigning value to it. For example, if you declare an *int value x*, and after this, if we try to print it or use it, then it will through an error because we haven’t assigned any value to it. But in other programming languages like CPP, it is possible.
- To get the datatype class of any variable -
  `System.out.println(((Object) x).getClass());`
- To get just the datatype of any variable - use getsimpleName()
  `System.out.println(((Object) x).getClass().getSimpleName());`

- **Byte** is a datatype in JAVA, it stores integer value only but it takes only 1 byte of space in RAM, meaning with this datatype we can store values ranging from -128 to 127.
- **Addition Program**:
  ```
  int x=5;
  byte y=10;
  System.out.println(x+y);
  ```
  It will print x+y and the datatype of (x+y) will be an integer. because x is *int* and y is *byte* hence x+y would be of int datatype, We always take bigger datatype(here int is a bigger datatype in comparison to “byte”).

- Here, If we try to use the “+” operator between two-character variables, then the integer value will be returned.
  ```
  char i='A';
  char j='B';
  System.out.println(i+j);    //-> 131
  ```
  Here the output will be 131, char datatype in java comes from the number family so when we do an arithmetic expression between two characters, then “char” automatically does type casting to integer and then start behaving like int.

- **To add two strings**:
  ```
  String i = "Hello";
  String j= " world";
  System.out.println(i+j);
  ```

- **Polymorphism** - The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. So for example in the above example, we can see, everywhere we are using the “+” operator but its behavior changes with the datatype, If we use the “+” operator between integers then it just adds them, but if we do between two strings it just shows both the strings together and so on.
- **Type coercion** is the automatic or implicit conversion of values from one data type to another (such as strings to integers), for example when we try to add int data with byte data then their result will automatically become “int”
- In JAVA we declare a char value with a single inverted comma(‘’) and for string, we use the double inverted comma(“”), the vice-versa will not work
  char i =’ a’;
  String j=” hello”;

- Take **input** from standard input device
  ```
  import java.util.Scanner;
  Scanner sc= new Scanner(System.in);
  int x=sc.nextInt();
  System.out.println(x);
  ```

  Explanation - Now *Import java.util.Scanner* will import the Scanner class. And then

Scanner sc =new Scanner(System.in) here, with the *system.in* we are defining that we will be using a standard input device that is the keyboard, Then we are asking the *Scanner class* to keep on scanning the keyboard, now with *int x=sc.nextInt()* , we are asking to read the inputs scanned by the scanner class as an integer and store it in RAM with an integer datatype.

- Like nextInt we have a lot of more functions like next(), nextLine() which reads the data from the keyboard as a String,
- `String x =sc.next();`
  It will just take the input as a string but it will only read until we do not press a spaceBar(it just takes one word as input), So if we want to continue reading as a string then use - `String x =sc.nextLine();`
  Now, this has the capability to read the complete sentence.

- Now for taking the character input from the user -
  `char x =sc.next().charAt(0);`
  Now it will read just the first character no matter how much big string the user has typed.

- useRadix(2) - it will now Scan the input as a binary value(so we can type only 0,1 as the input) and then will convert it into an integer.
  ```
  sc.useRadix(2);
  int x =sc.nextInt() ;
  ```

- It will not print 5+6=11, It will print 56(Remember String+Int =String) In starting we have added a String, and after this, we are adding an int value using the + operator, now adding them will return a new string.
  ```
  int l=5;
  int m=6;
  System.out.println("    "+l+m); //will print 56

  System.out.println((l+m)/2);// It will print 5   - BODMAS Rule
  ```