**Core Java**

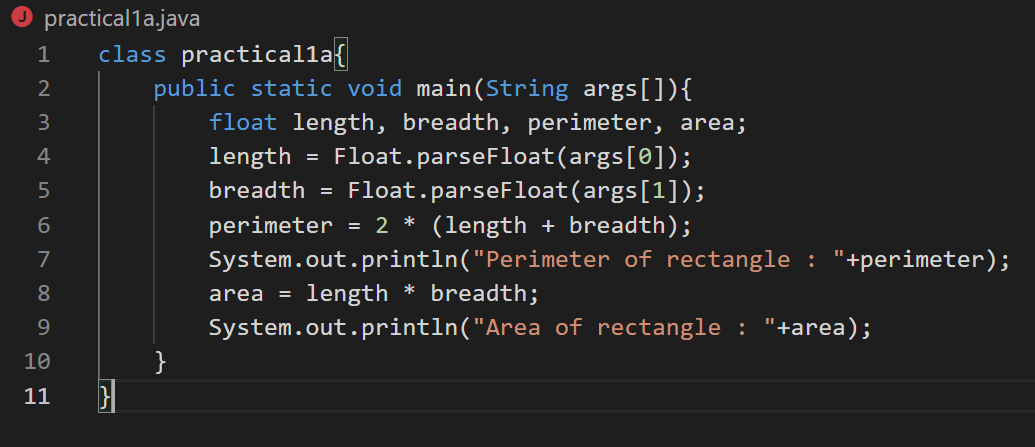
**Practical 1 : Java Data Types**

1. **Aim :** Write a program to accept length and breadth as command line arguments and calculate area and perimeter of rectangle.

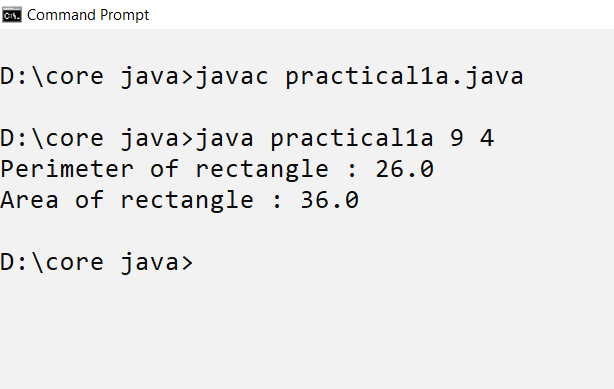
**Theory :**

1. class keyword is used to declare a class in Java.
2. public keyword is an access modifier that represents visibility. It means it is visible to all.
3. The main () method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
4. void is the return type of the method. It means it doesn't return any value.
5. String[] args or String args[] is used for command line argument.
6. System.out.println() is used to print statement. Here, System is a class, out is an object of the Print Stream class, println () is a method of the Print Stream class.
7. The java command-line argument is an argument i.e., passed at the time of running the java program.
8. The arguments passed from the console can be received in the java program and it can be used as an input.
9. The float data type can store fractional numbers. We should end the value with an "f"

**Code :**



**Output :**

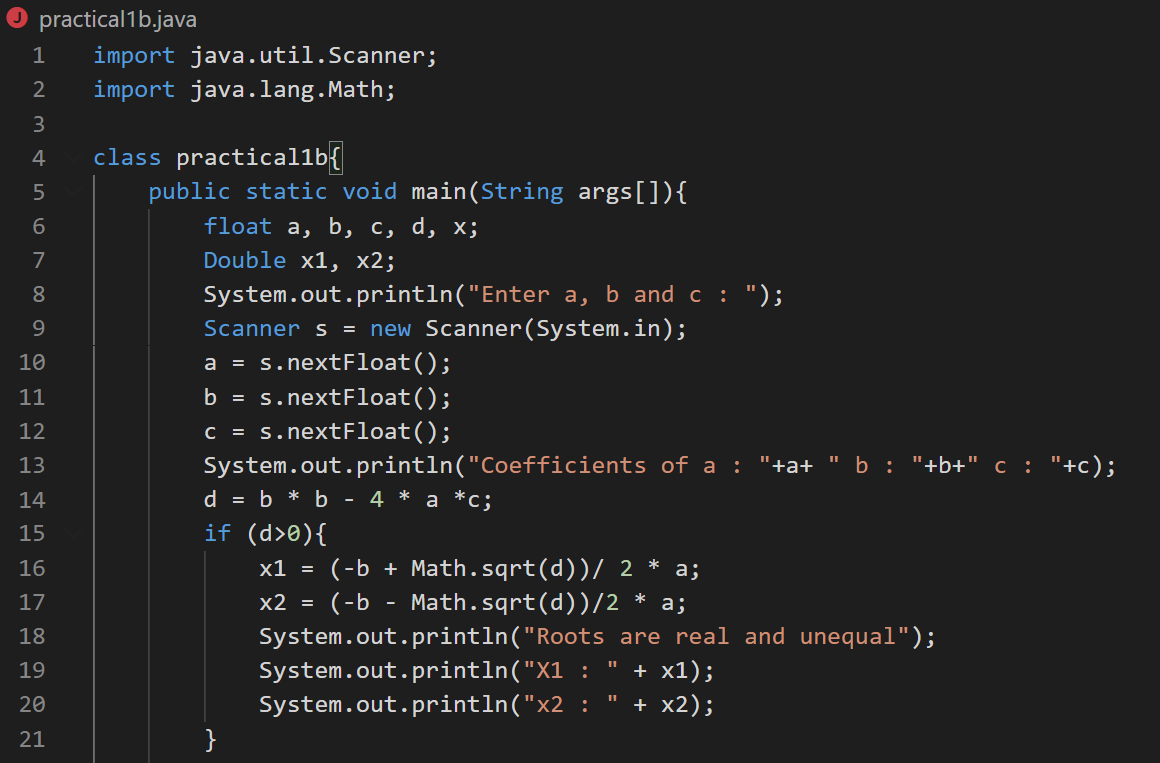


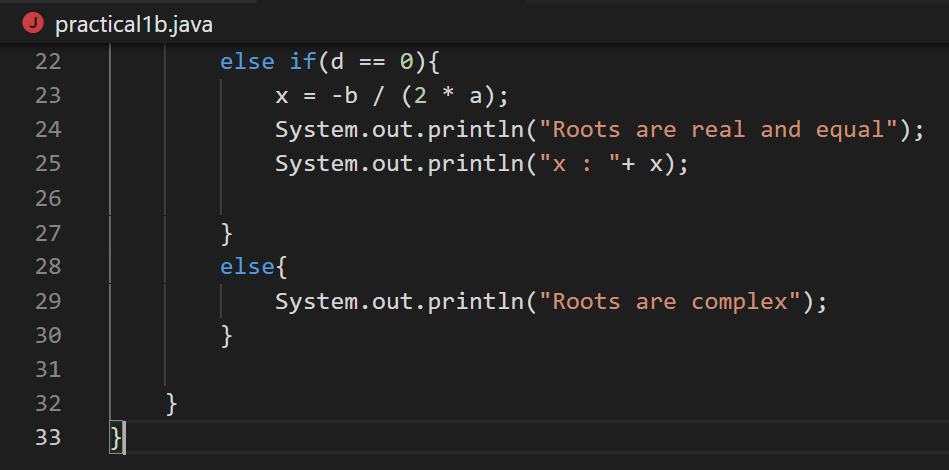
1. **Aim :** Write a program to accept integer values for a, b and c which are coefficients of quadratic equation and find the solution of quadratic equation.

**Theory**

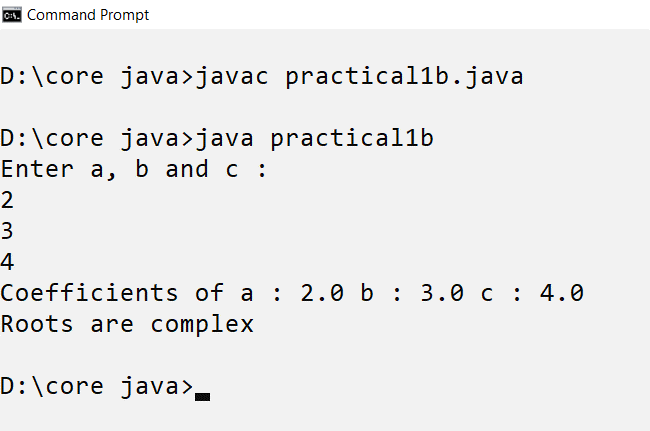
1. The java Math class has many methods that allows us to perform mathematical taskas on numbers
2. The scanner class is used to get user input and it is found in the java,util package.
3. To use the Scanner class, create an object of the class and use any of the available methods found in the scanner class documentation. In our example we will use the nextFloat() method, which is used to read float numbers.

**Code :**





**Output :**



**Practical 2 : Use of Operators**

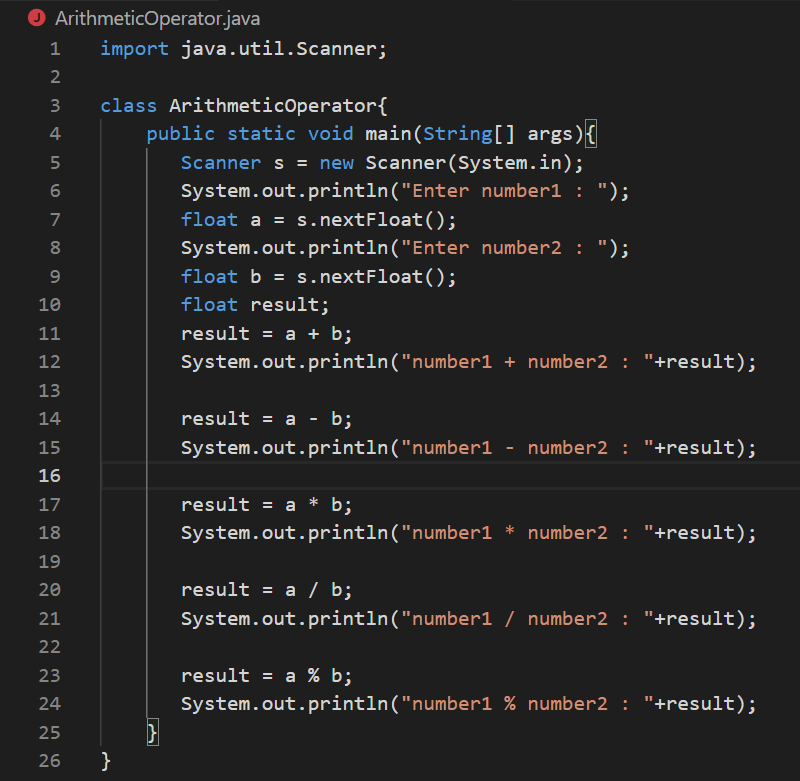
1. **Aim :** Demostrating the use of various types of operators supported by java

**Theory :**

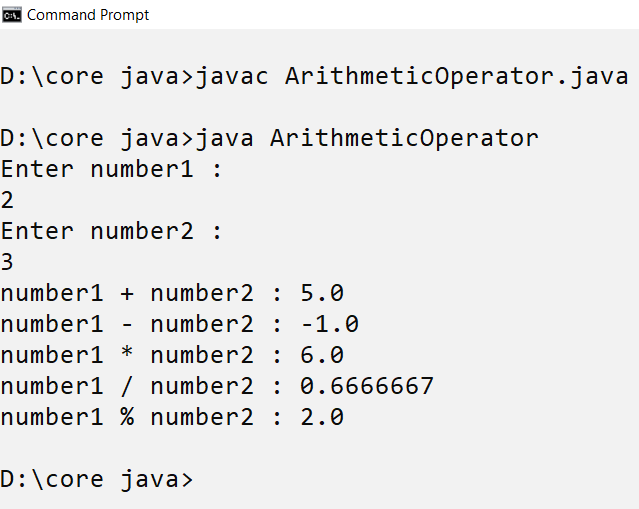
|  |  |
| --- | --- |
| Operators | result |
| Arithmetic Operator | |
| + | Addition of two numbers |
| - | Subtraction of two numbers |
| \* | Multiplication of two numbers |
| / | Division of two numbers |
| % | Divides two numbers and returns the remainder |
| Increment Decrement Operator | |
| x++ | Post Increment |
| ++x | Pre Increment |
| x-- | Post Decrement |
| --x | Pre Decrement |
| Logical Operator | |
| &&(Logical AND) | true only if both expression1 and expression2 are true |
| ||(Logical OR) | true if either expression1 or expression2 is true |
| !(Logical NOT) | true if expression is false and vice versa |
| Relational Operator | |
| == | Is equal to |
| != | Not equal to |
| > | Greater than |
| >= | Greater than equal to |
| < | Less than |
| <= | Less than equal to |
| Instance Of Operator | |
| instanceof | instanceof is a keyword that is used for checking if a reference variable is containing a given type of object reference or not. |

1. **Arithmetic Operator**

**Code :**

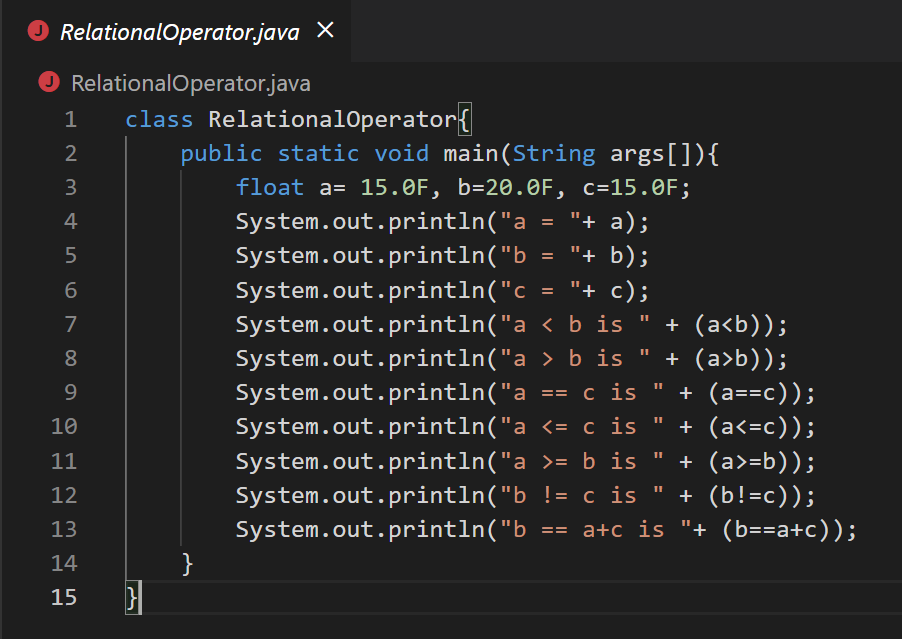


**Output :**

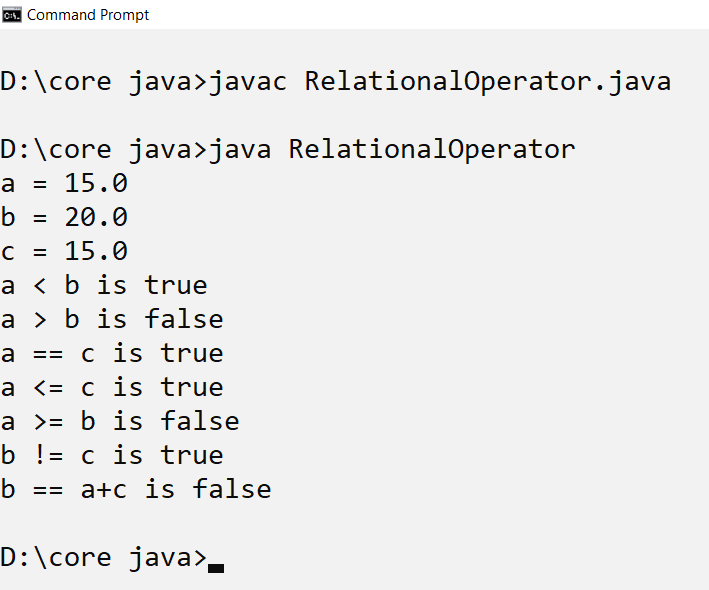


1. Relational Operator

**Code :**

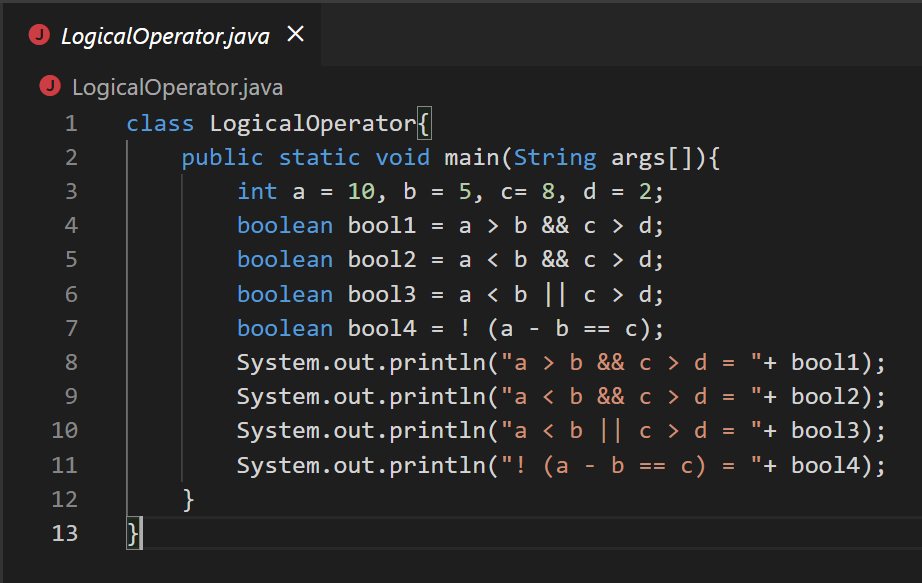


**Output :**

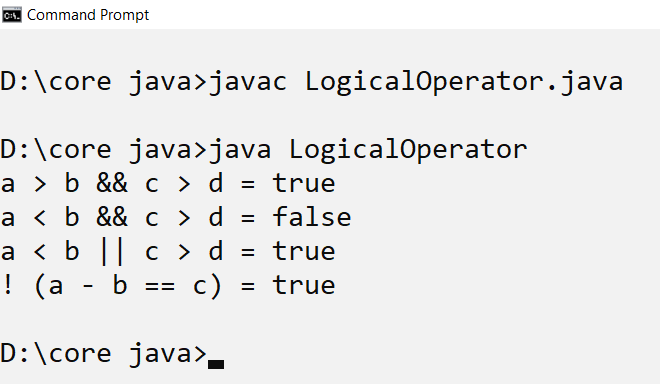


1. Logical Operator

**Code :**

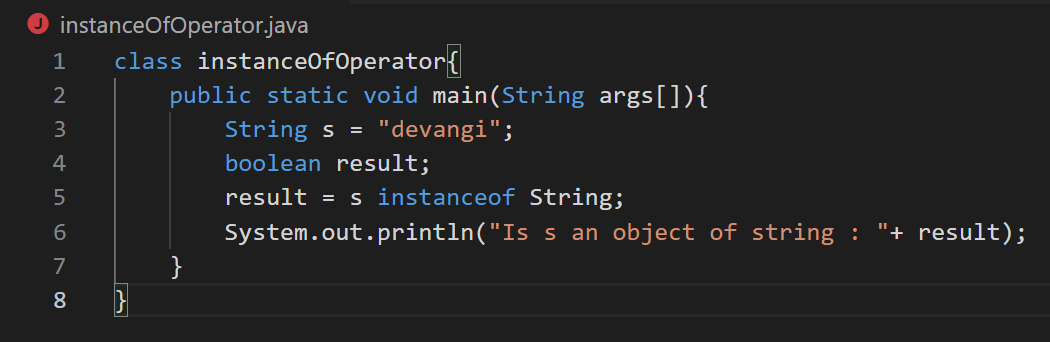


**Output :**

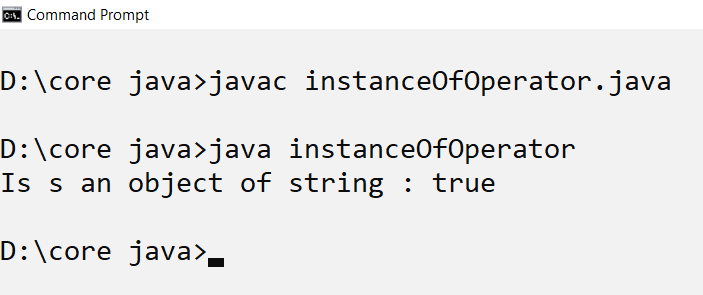


1. InstanceOfOperator :

**Code :**

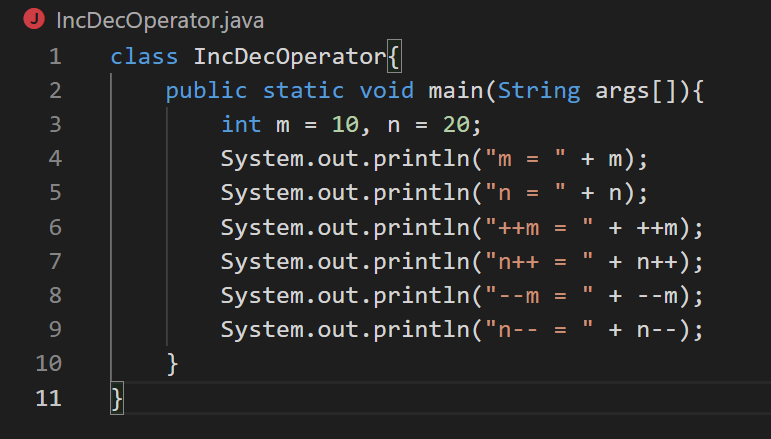


Output :

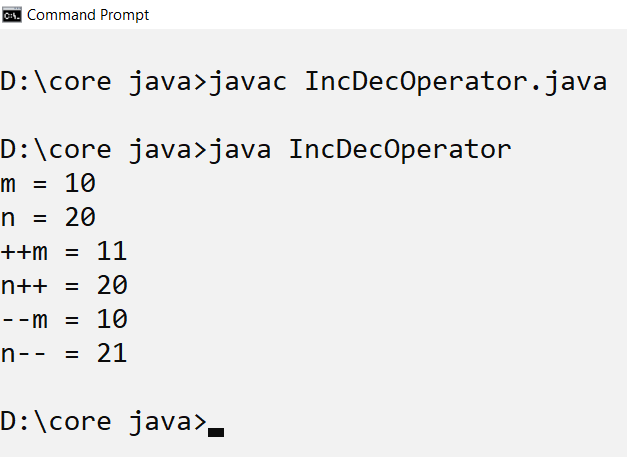


1. Increment Decrement operator

**Code :**



**Output :**

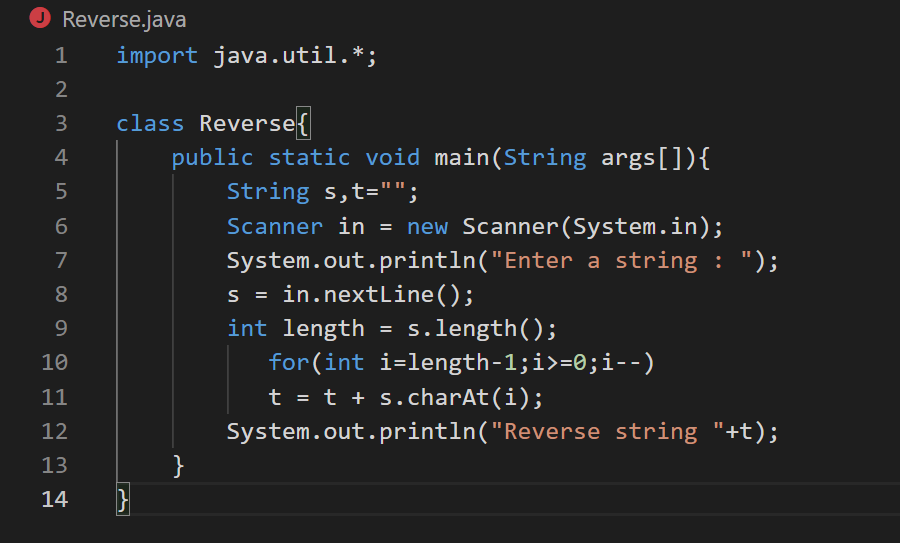


1. **Aim :** Program to a reverse string.

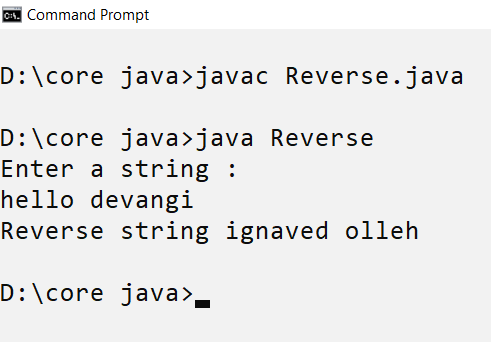
**Theory :**

1. The length() returns the length of a string object
2. nextLine() method, which is used to read Strings.
3. For Loop
4. Initialization: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.
5. Condition: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.
6. Increment/Decrement: It increments or decrements the variable value. It is an optional condition.

**Code :**



**Output :**



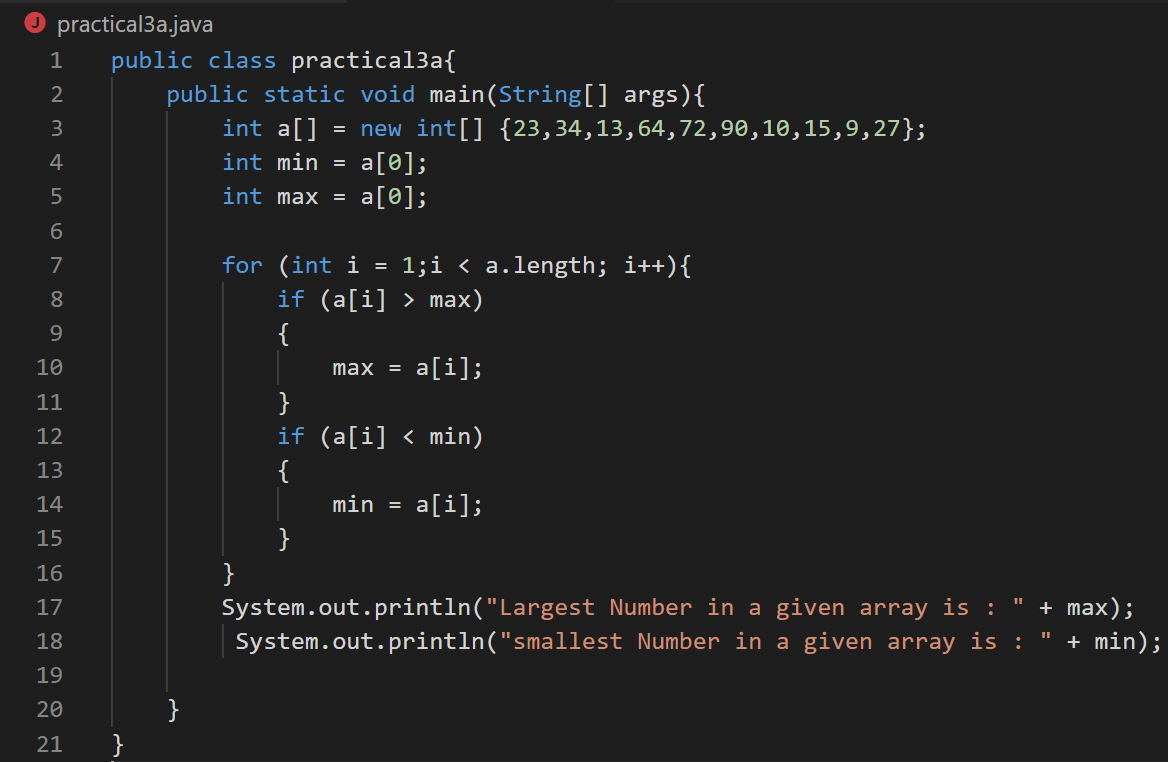
**Practical 3 : Use of control statement and iterators**

1. **Aim :** Program to find smallest and largest elwmwnt from the array.

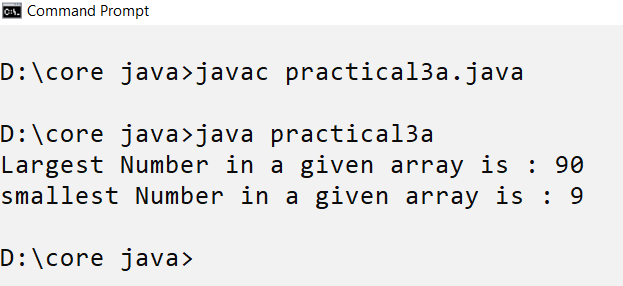
**Theory :**

1. **Java array** is an object which contains elements of a similar data type.
2. Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.
3. if statement to specify a block of Java code to be executed if a condition is true.

**Code :**



**Output :**

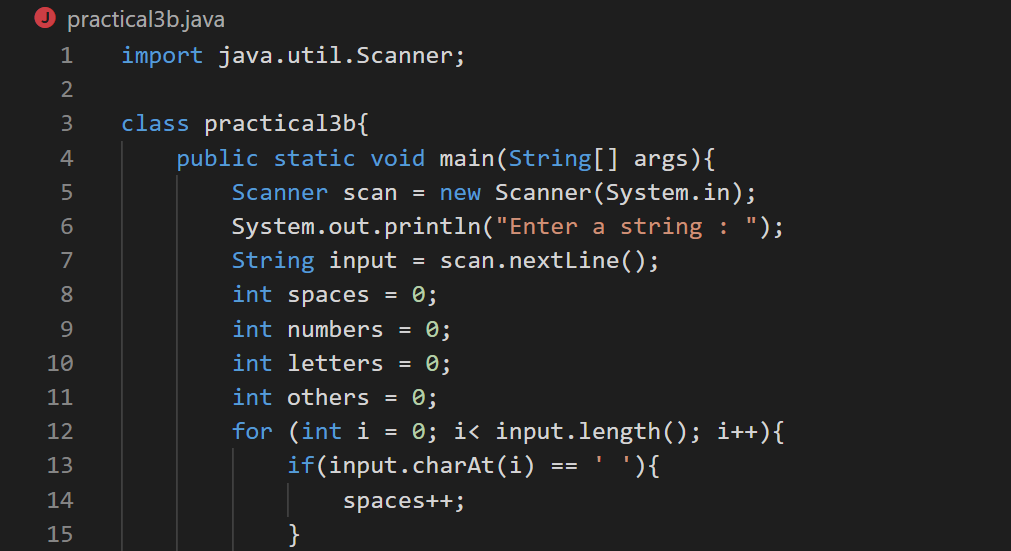


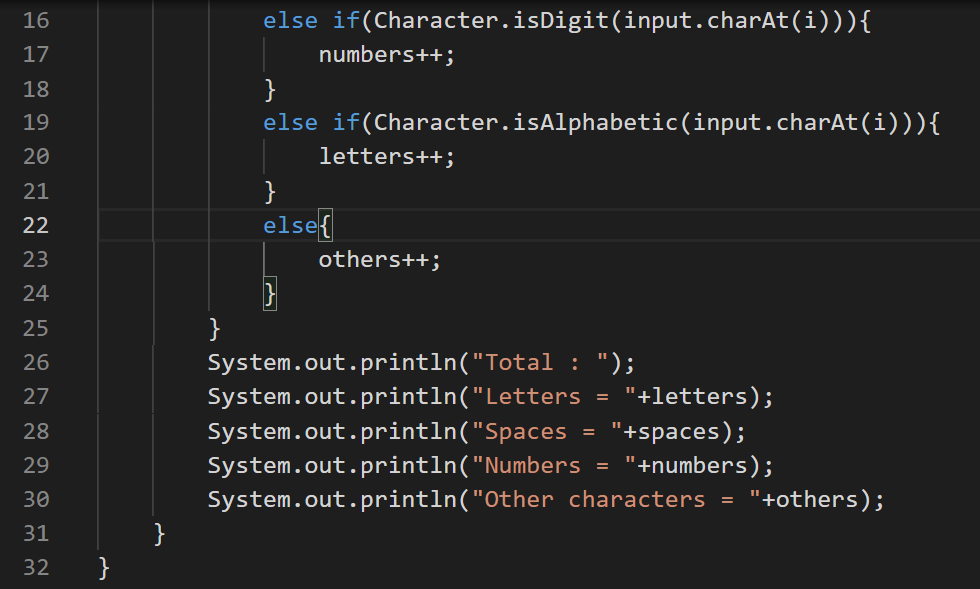
1. **Aim :** Program to count the letters, spaces, numbers and other characters of an input string.

**Theory :**

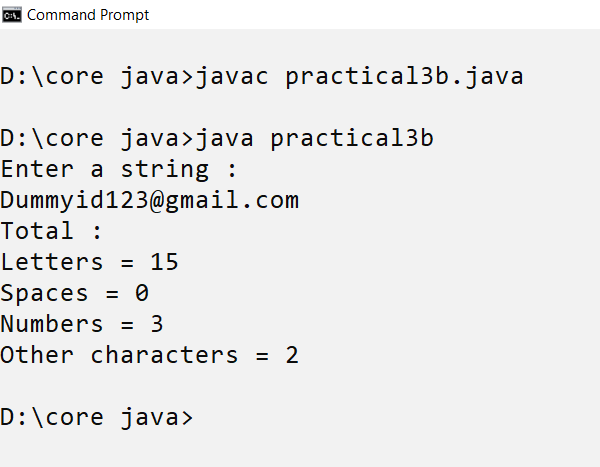
1. if statement to specify a block of Java code to be executed if a condition is true.
2. else statement to specify a block of code to be executed if the condition is false.
3. else if statement to specify a new condition if the first condition is false.
4. charAt() method returns a character at a specific index position in a string.
5. isDigit () method is used to determine whether the specified character is a digit.
6. isAlphabetic()method checks whether the argumented character is an alphabet or not.

**Code :**





**Output :**



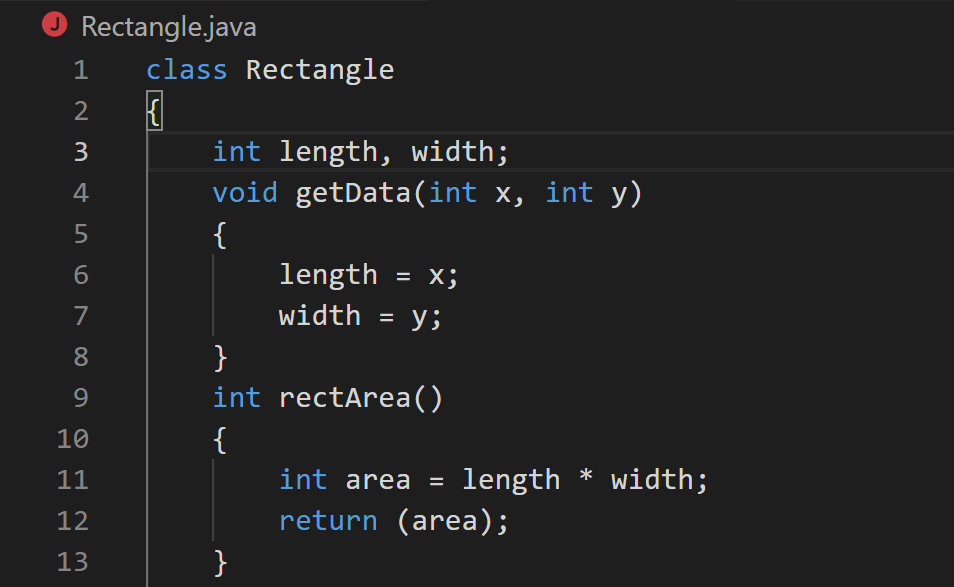
**Practical 4 : Using Classes and Objects**

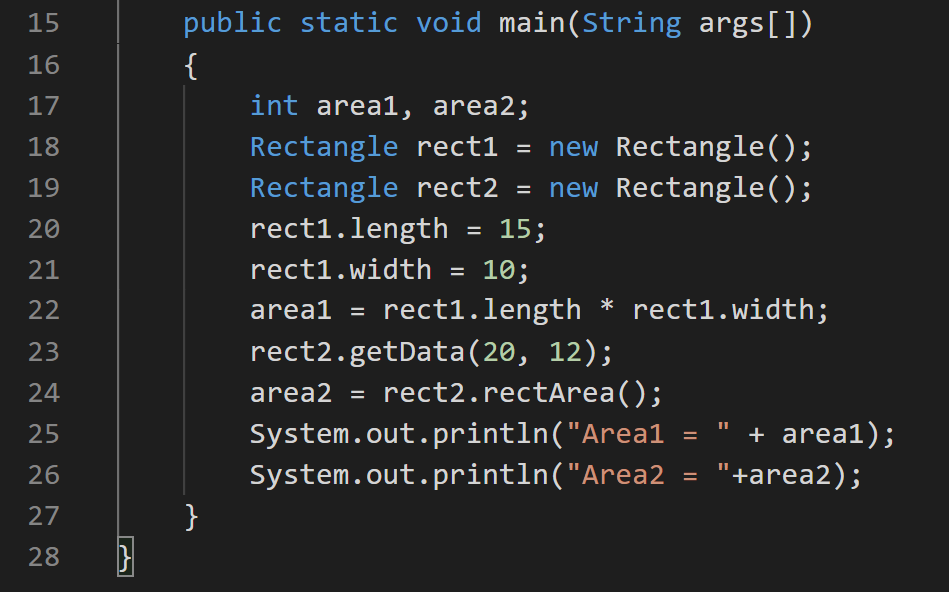
1. **Aim :** Designing a class in java which includes instance methods and instance variables and initialize them by creating object.

**Theory :**

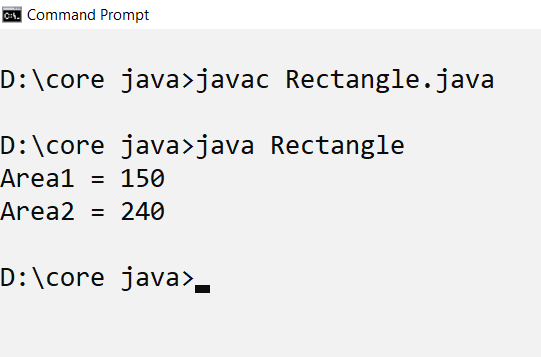
1. First we will create a class called Rectangle in which we will create main method, define the variable length and breadth, and the method rectArea.
2. Inside the main method, the object of the rectangle is created using which we can call assign the values to the variable, and after assigning the values the methods will be called to calculate the Area Rectangle.
3. length and breadth that we had passed from the main method using the object of that class.
4. Return keyword is used to exit from the method with or without value.

**Code :**





**Output :**

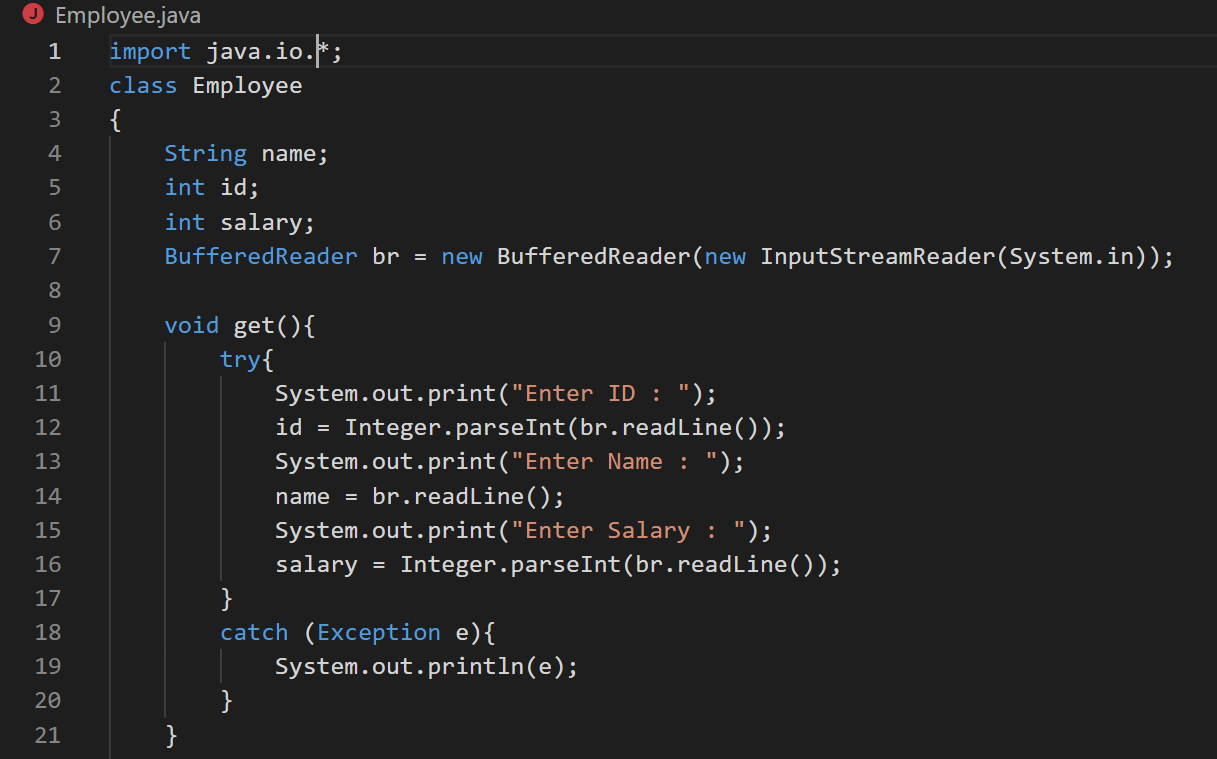


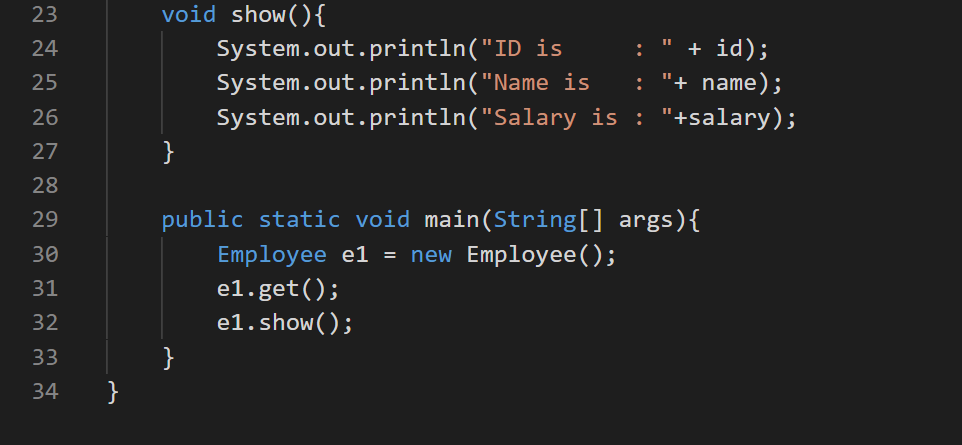
1. **Aim :** Demonstrating use of constructor in java.

**Theory :**

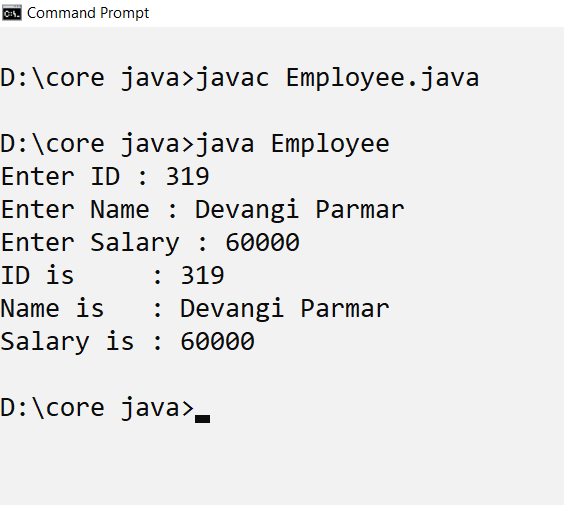
1. Import.java.io.\* : Import basically means we are adding a Library/Package full of classes and methods that are premade for ease of use. Java.IO is standard for Java Input/Output. The asterisk on the end means we are importing everything in the Java.IO Package.
2. Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method.
3. InputStreamReader class is a bridge from byte streams to character streams.It reads bytes and decodes them into characters using a specified charset.
4. Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.
5. Java catch block is used to handle the Exception by declaring the type of exception within the parameter.

**Code :**





**Output :**



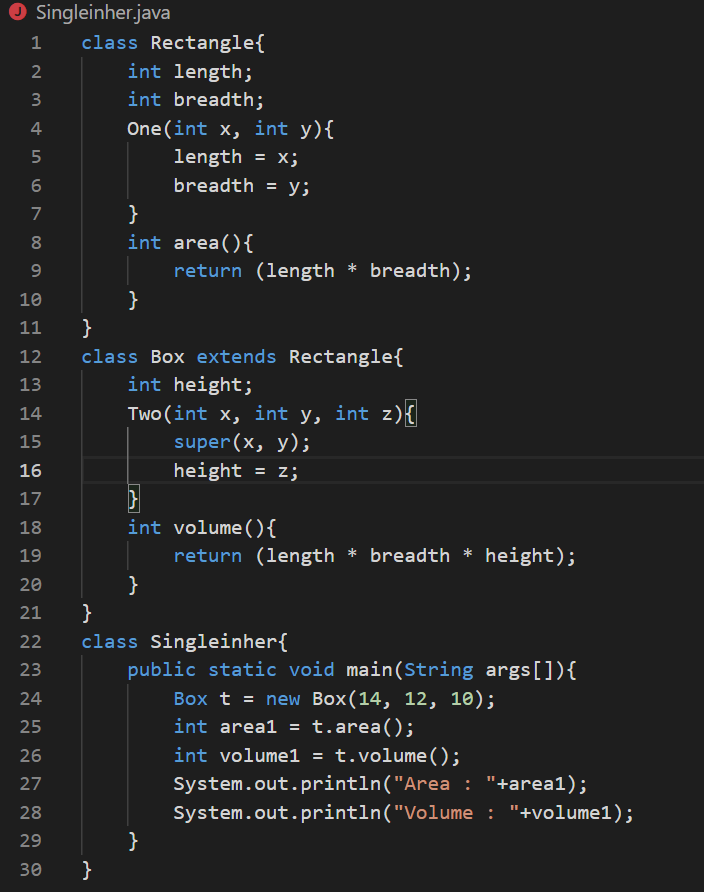
**Practical 5 : Inheritance**

1. **Aim :** Program to implement single level inheritance.

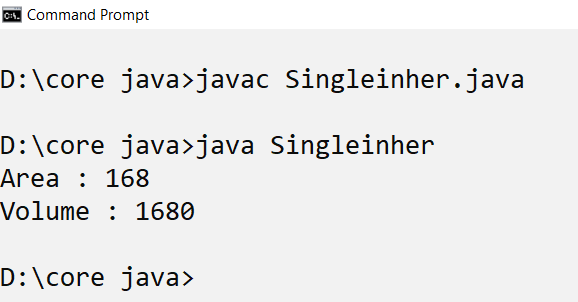
**Theory:**

1. **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object.
2. The class whose features are inherited is known as superclass(or a base class or a parent class).
3. The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class).
4. In single inheritance, subclasses inherit the features of one superclass.

**Code :**



**Output :**

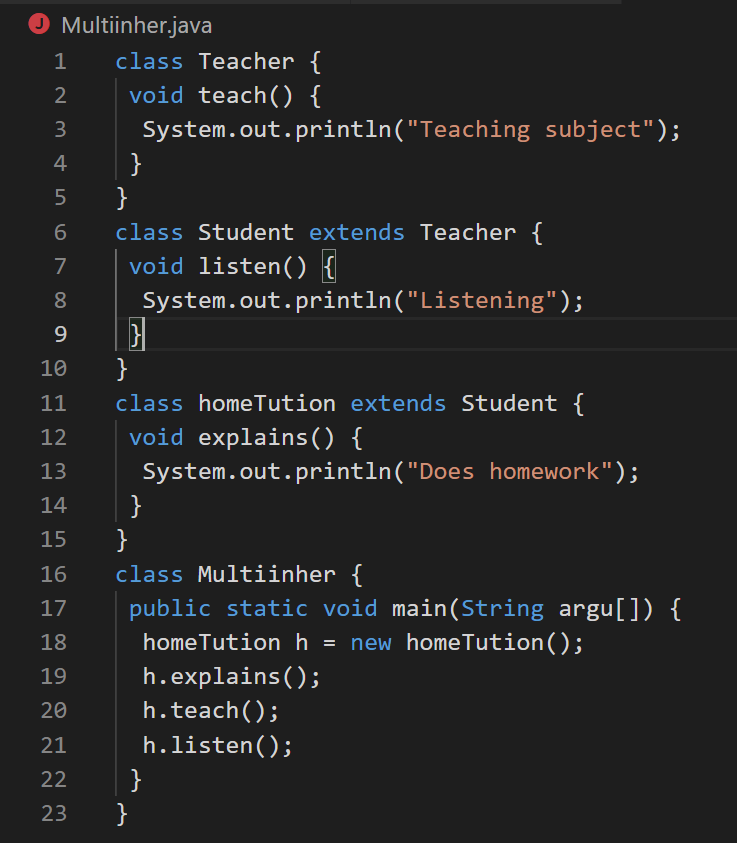


1. **Aim :** Program to implement Multilevel inheritance.

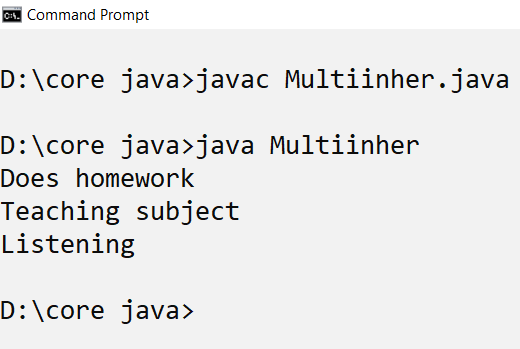
**Theory :**

1. In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.

**Code :**



**Output :**

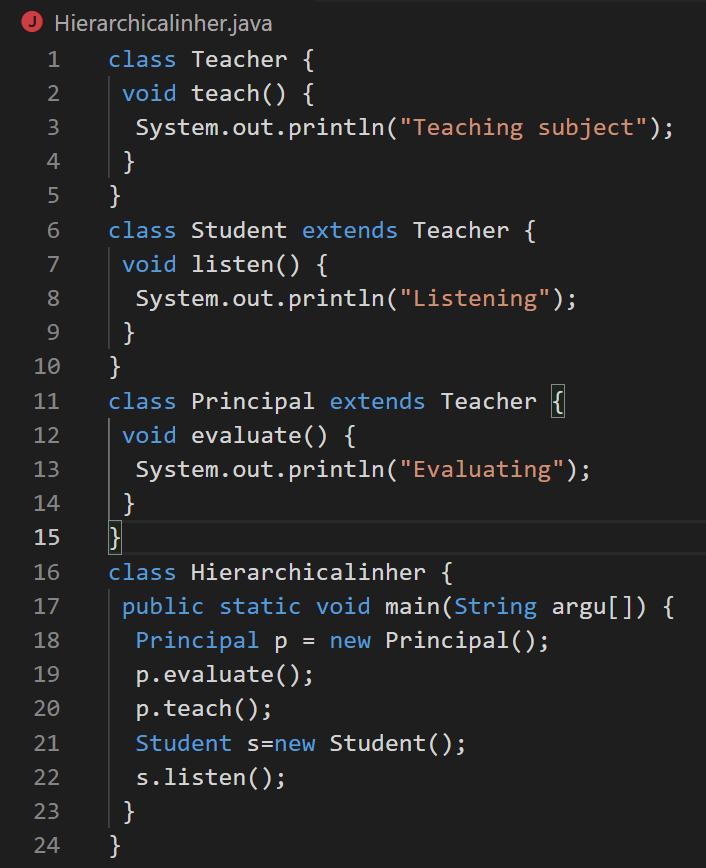


1. **Aim :** Program to implement hierarchical inheritance.

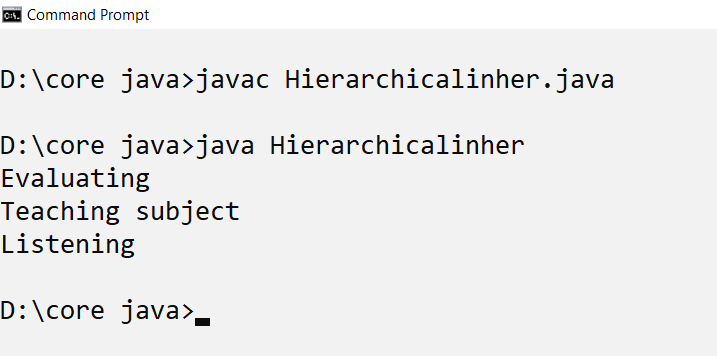
**Theory :**

1. In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass.

Code :



**Output :**

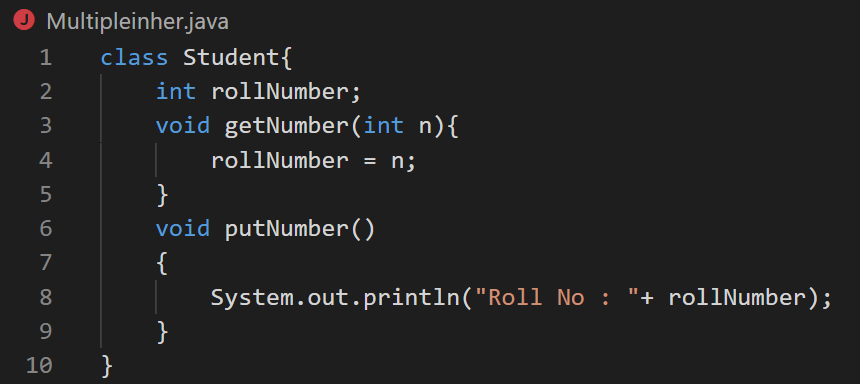


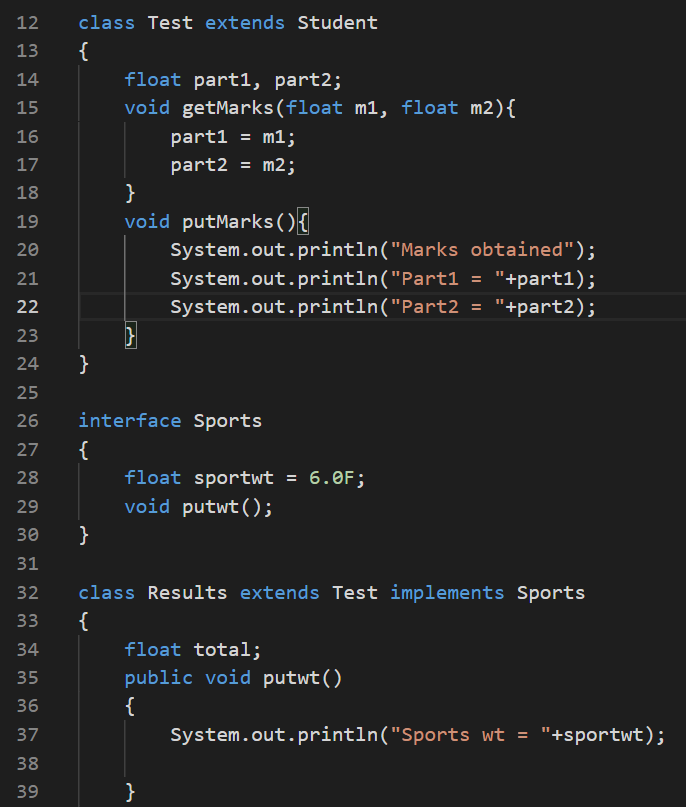
1. **Aim :** Program to implement Multiple inheritance.

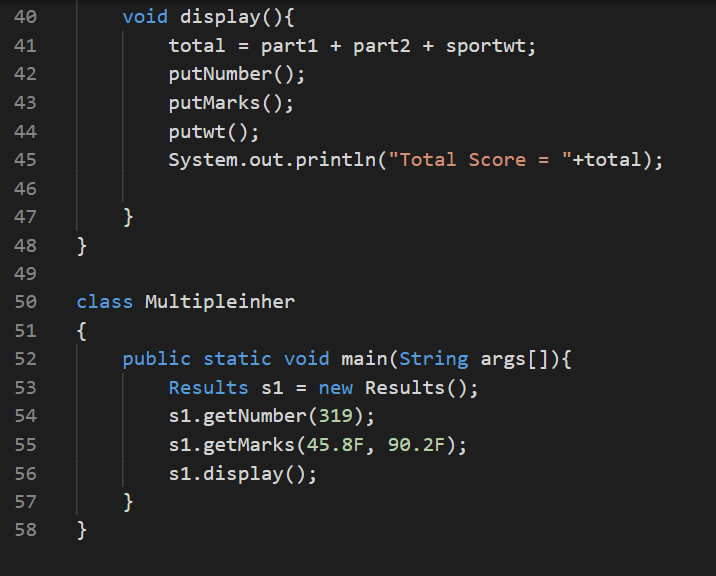
**Theory :**

1. In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.
2. In java, we can achieve multiple inheritances only through Interfaces.
3. An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.
4. The interface in Java is a mechanism to achieve abstraction.

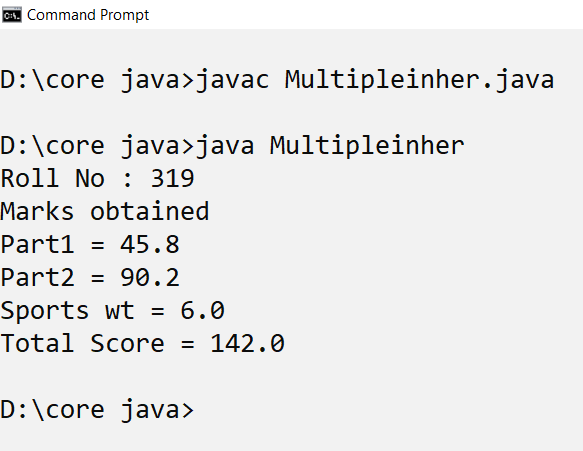
Code :







**Output :**



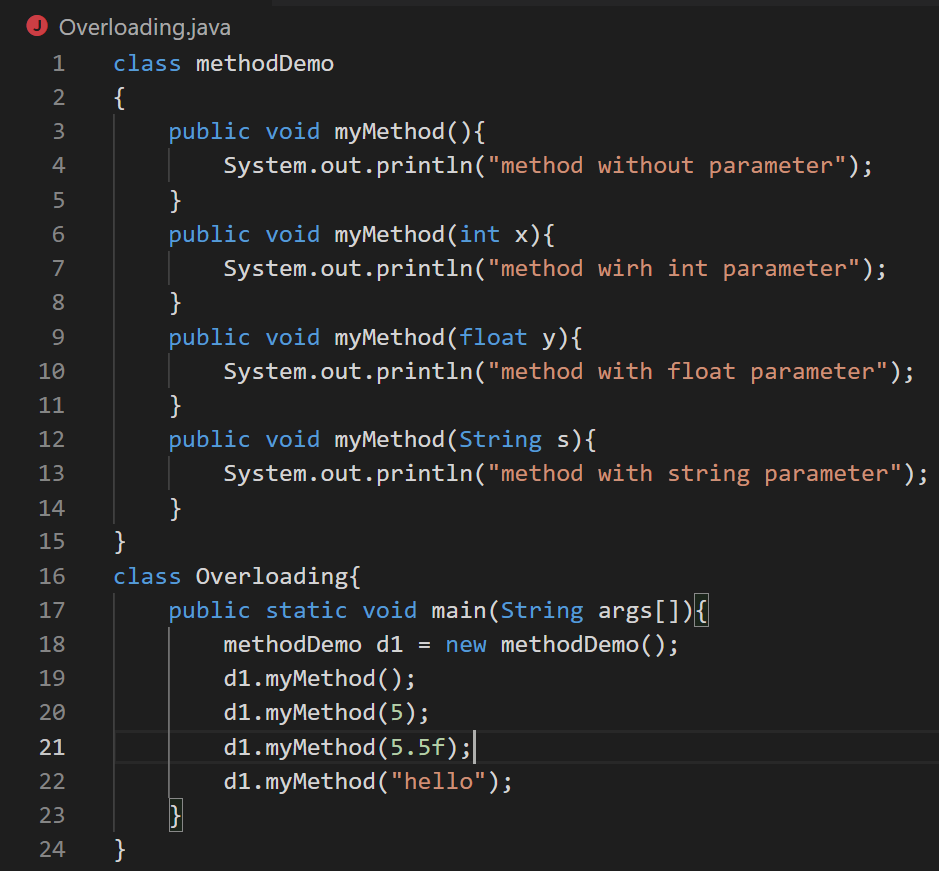
**Practical 6 : Polymorphism**

1. **Aim :** Program to implement method overloading.

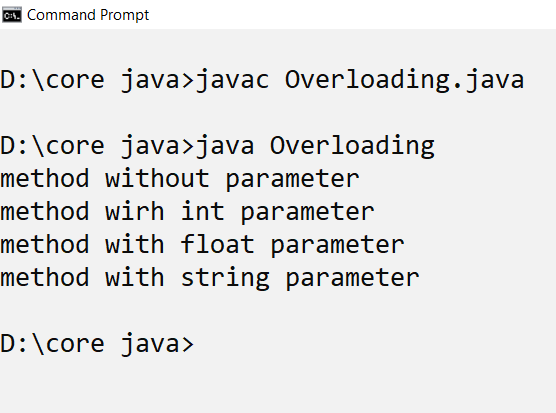
**Theory :**

1. Polymorphism means the ability to take various forms. Polymorphism allows us to define methods in the child class withthe same name as defined in their parent class.
2. If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

Code :



Output:

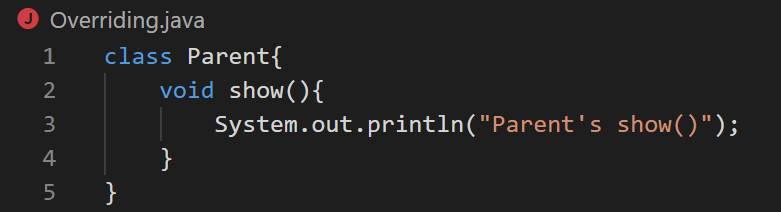


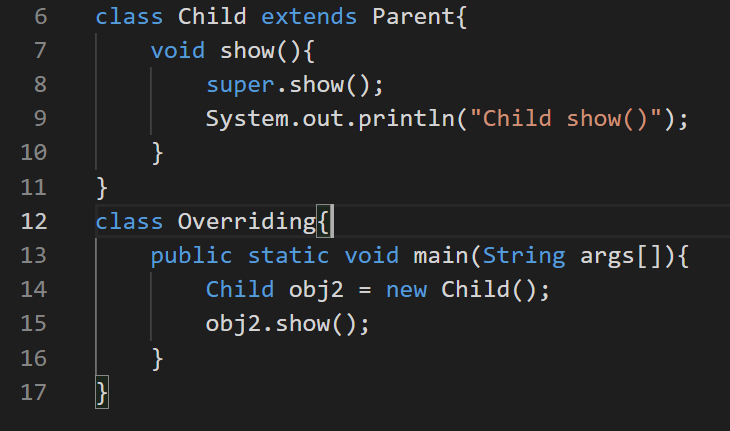
1. **Aim :** Program to implement Method Overriding.

**Theory :**

1. If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**. Method overriding is used to provide the specific implementation of a method which is already provided by its superclass. Method overriding is used for runtime polymorphism

**Code :**





**Output :**

