Practical no. 2

Practical no. 2 FS19CO042

Aim: Perform following programs.

2.1 Write a program to print "Hello World".

2.2 Write a program to print addition of two integers.

2.3 Write a program to convert a numeric string into int.

2.4 Write a program to print addition of two integers input from command line arguments.

2.5 Write a program to take two integers from command line, subtract the smaller number from the greater and print the result.

2.6 Write a program to take n integers from command line and print their sum of product (product of first number and last number added to product of second number and second last number and so on).

2.7 Consider any two integers. Write a program to print sum of their squares.

2.8 Write a program to find square root of a given positive integer using Heron's method to find square root.

2.9 Write a program to sort and print the names of students taken from command line in alphabetical order.

2.10 Write a program to print total numbers of vowels and consonants in a given string.

2.11 Given two English words, write a program to check if the first word is anagram of the second word. (An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. (Example: Anagram of TOM MARVOLO RIDDLE is I AM LORD VOLDEMORT.)

2.12 Write a program to print a missing number in a sorted integer array.

2.13 Write a program to find all the pairs of numbers on an integer array whose sum is equal to a given number.

Tool used: Editor (Notepad/Intellij IDE), JDK and JRE

Theory:

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.
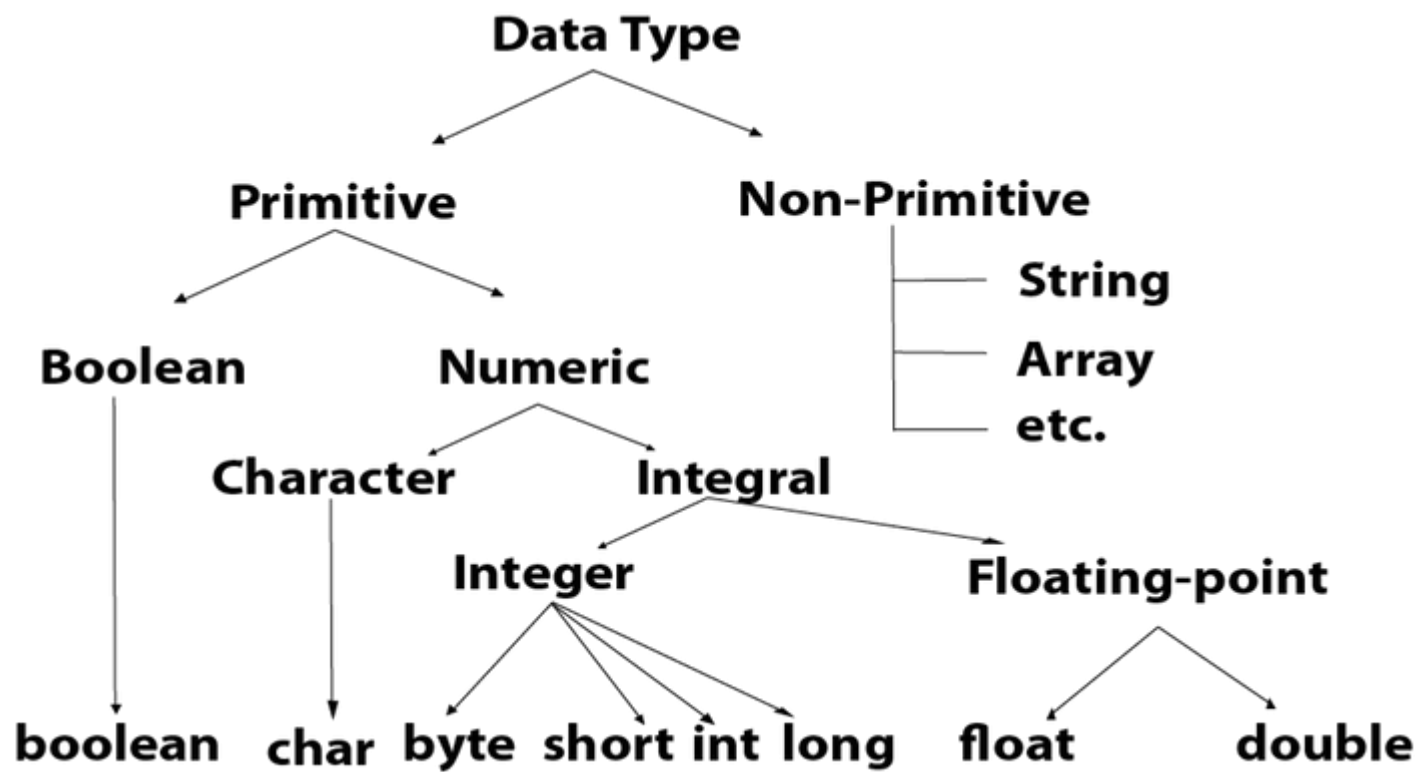
Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type

- o   long data type
- o   float data type
- o   double data type

Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: Boolean one = false

Byte Data Type

The byte data type is an example of primitive data type. It isan 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0.

The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

Example: byte a = 10, byte b = -20

Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example: short s = 10000, short r = -5000

Int Data Type

The int data type is a 32-bit signed two's complement integer. Its value-range lies between - 2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive). Its minimum value is - 2,147,483,648and maximum value is 2,147,483,647. Its default value is 0.

The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

Example: int a = 100000, int b = -200000

Long Data Type

The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 - 1)(inclusive). Its minimum value is - 9,223,372,036,854,775,808and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example: long a = 100000L, long b = -200000L

Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point.Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Example: float f1 = 234.5f

Double Data Type

The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Example: double d1 = 12.3

Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

Example: char letterA = 'A'

Java Operators

In this tutorial, you'll learn about different types of operators in Java, their syntax and how to use them with the help of examples.

Operators are symbols that perform operations on variables and values. For example, `+` is an operator used for addition,

while `*` is also an operator used for multiplication.

Operators in Java can be classified into 5 types:

1. Arithmetic Operators

2. Assignment Operators

3. Relational Operators

4. Logical Operators

5. Unary Operators

6. Bitwise Operators

---

1. Java Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on variables and data. For example, a + b;

Here, the `+` operator is used to add two variables `a` and `b`. Similarly, there are various other arithmetic operators in Java.

Operator	Operation
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo Operation (Remainder after division)

2. Java Assignment Operators

Assignment operators are used in Java to assign values to variables. For example,

int age;

age = 5;

Here, `=` is the assignment operator. It assigns the value on its right to the variable on its left. That is, **5** is assigned to the variable `age`.

Let's see some more assignment operators available in Java.

Operator	Example	Equivalent to
`=`	`a = b;`	`a = b;`
`+=`	`a += b;`	`a = a + b;`
`-=`	`a -= b;`	`a = a - b;`
`*=`	`a *= b;`	`a = a * b;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`a %= b;`	`a = a % b;`

3. Java Relational Operators

Relational operators are used to check the relationship between two operands. For example,

// check is a is less than b

a < b;

Here, `>` operator is the relational operator. It checks if `a` is less than `b` or not.

It returns either `true` or `false`.

Operator	Description	Example
`==`	Is Equal To	`3 == 5` returns **false**
`!=`	Not Equal To	`3 != 5` returns **true**
`>`	Greater Than	`3 > 5` returns **false**
`<`	Less Than	`3 < 5` returns **true**
`>=`	Greater Than or Equal To	`3 >= 5` returns **false**
`<=`	Less Than or Equal To	`3 <= 5` returns **false**

4. Java Logical Operators

Logical operators are used to check whether an expression is `true` or `false`. They are used in decision making.

Operator	Example	Meaning
&& (Logical AND)	expression1 **&&** expression2	`true` only if both `expression1` and `expression2` are `true`
|| (Logical OR)	expression1 **||** expression2	`true` if either `expression1` or `expression2` is `true`
! (Logical NOT)	**!**expression	`true` if `expression` is `false` and vice versa

5. Java Unary Operators

Unary operators are used with only one operand. For example, `++` is a unary operator that increases the value of a variable by **1**. That is, `++5` will return **6**.

Different types of unary operators are:

Operator	Meaning
`+`	**Unary plus**: not necessary to use since numbers are positive without using it
`-`	**Unary minus**: inverts the sign of an expression
`++`	**Increment operator**: increments value by 1
`--`	**Decrement operator**: decrements value by 1
`!`	**Logical complement operator**: inverts the value of a boolean

Increment and Decrement Operators

Java also provides increment and decrement operators: `++` and `--` respectively. `++` increases the value of the operand by **1**, while `--` decrease it by **1**. For example,

int num = 5;

// increase num by 1

++num;

Here, the value of `num` gets increased to **6** from its initial value of **5**.

Example 5: Increment and Decrement Operators

In the above program, we have used the ++ and -- operator as **prefixes (++a, --b)**. We can also use these operators as **postfix (a++, b++)**.

There is a slight difference when these operators are used as prefix versus when they are used as a postfix.

6. Java Bitwise Operators

Bitwise operators in Java are used to perform operations on individual bits. For example,

Bitwise complement Operation of 35
35 = 00100011 (In Binary)
~ 00100011
_____
  11011100 = 220 (In decimal)

Here, `~` is a bitwise operator. It inverts the value of each bit (**0** to **1** and **1** to **0**).

The various bitwise operators present in Java are:

Operator	Description
`~`	Bitwise Complement

<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR

These operators are not generally used in Java. To learn more, visit Java Bitwise and Bit Shift Operators.

Other operators

Besides these operators, there are other additional operators in Java.

Java instanceof Operator

The `instanceof` operator checks whether an object is an instanceof a particular class.

Here, `str` is an instance of the `String` class. Hence, the `instanceof` operator returns `true`.

Java Ternary Operator

The ternary operator (conditional operator) is shorthand for the `if-then-else` statement. For example,

variable = Expression ? expression1 : expression2

Here's how it works.

- If the Expression is true, expression1 is assigned to the variable.
- If the Expression is false, expression2 is assigned to the variable

Java Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with **square brackets**:

String[] cars;

We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

To create an array of integers, you could write:

int[] myNum = {10, 20, 30, 40};

Access the Elements of an Array

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

```
Example
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
// Outputs Volvo
```

Code:

2.1 Write a program to print "Hello World".

Code:
```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World !");
    }
}
```

Output:

2.2 Write a program to print addition of two integers.
Code:

```java
import java.util.Scanner;
public class add {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter first digit:");
        int a = scanner.nextInt();
        System.out.println("Enter second digit:");
        int b = scanner.nextInt();
        int c = a+b;
        System.out.println("Addition of two digits : " + c);
    }
}
```

Output:

2.3 Write a program to convert a numeric string into int.
Code:

```java
import java.util.Scanner;
public class string {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter string:");
        String s = scanner.nextLine();
        int number = Integer.parseInt(s);
        System.out.println("After parsing the string into Int: " + number);
    }
}
```

Output:

2.4 Write a program to print addition of two integers input from command line arguments.
Code:
```java
public class PR2_4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num1, num2, sum;
        num1 = Integer.parseInt(args[0]);
        num2 = Integer.parseInt(args[1]);
        sum = num1+num2;
        System.out.println(num1+" + "+num2+" = "+sum);
    }
}
```

Output:

2.5 Write a program to take two integers from command line, subtract the smaller number from the greater and print the result.
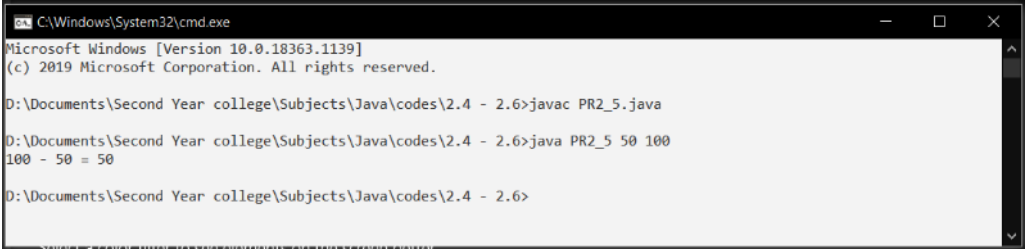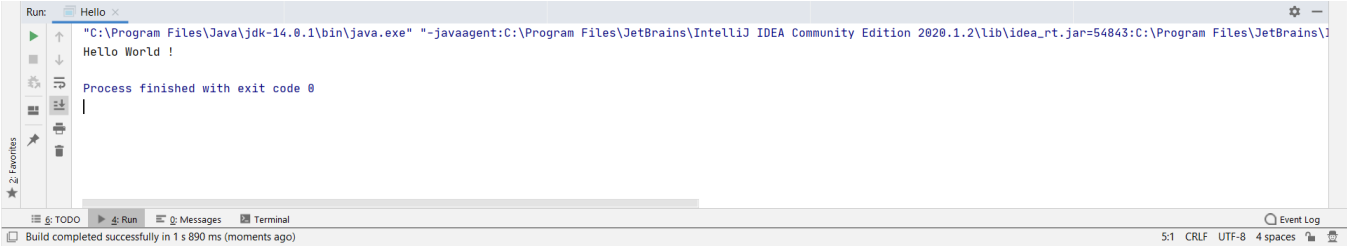Code:

```java
public class PR2_5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num1, num2;
        num1 = Integer.parseInt(args[0]);
        num2 = Integer.parseInt(args[1]);
        System.out.println(num1>num2 ? num1 +" - "+num2+" = "+(num1-num2) : num2 +" - "+num1+" = "+(num2-num1));
    }
}
```
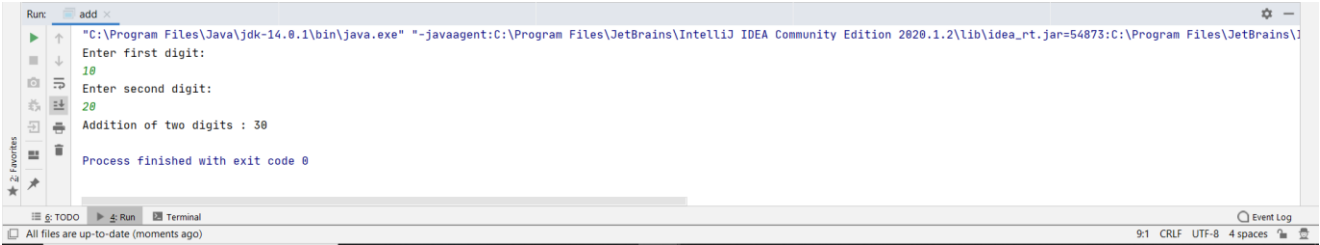
Output :

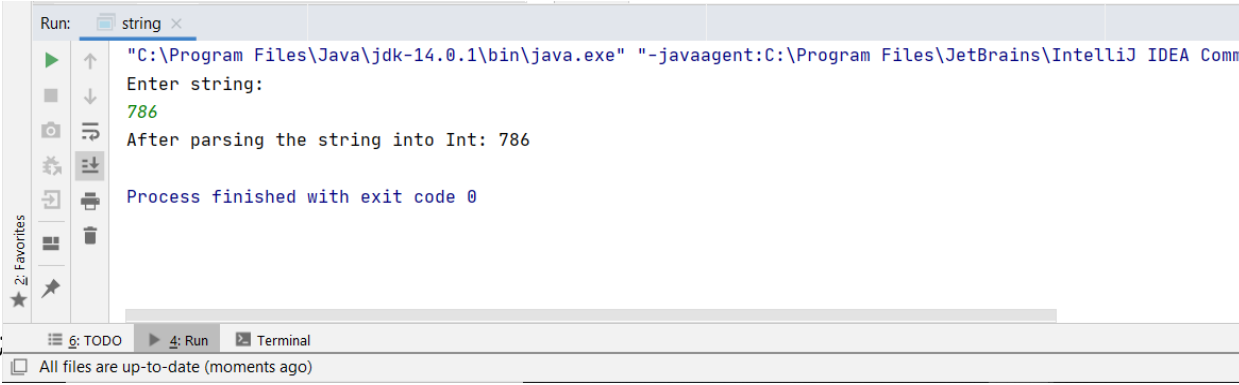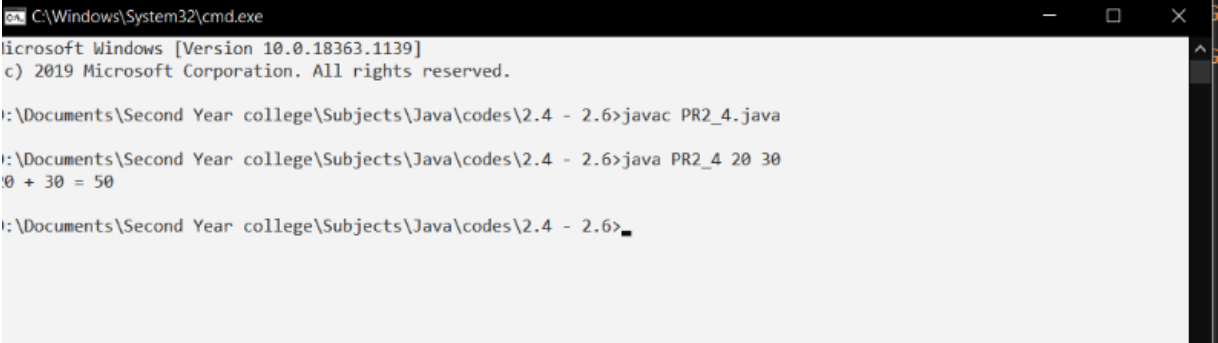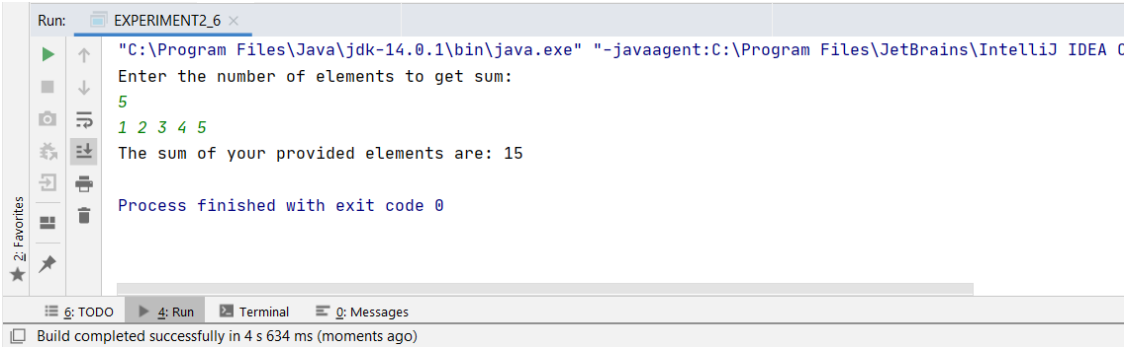**2.6 Write a program to take n integers from command line and print their sum of product (product of first number and last number added to product of second number and second last number and so on).**

Code:

```java
public class EXPERIMENT2_6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements to get sum:");
        int n = sc.nextInt();
        int [] arr = new int[n];
        int sum = 0;
        for(int k=0; k<arr.length; k++)
            arr[k] = sc.nextInt();
        for(int i=0; i<arr.length/2; i++){

            int result = arr[i] + arr[arr.length-1-i];
            sum += result;
        }
        if(n%2 != 0){
            int middleIndex = ((n-1)/2);
            sum += arr[middleIndex];
        }
        System.out.println("The sum of your provided elements are: "+sum);
    }
}
```
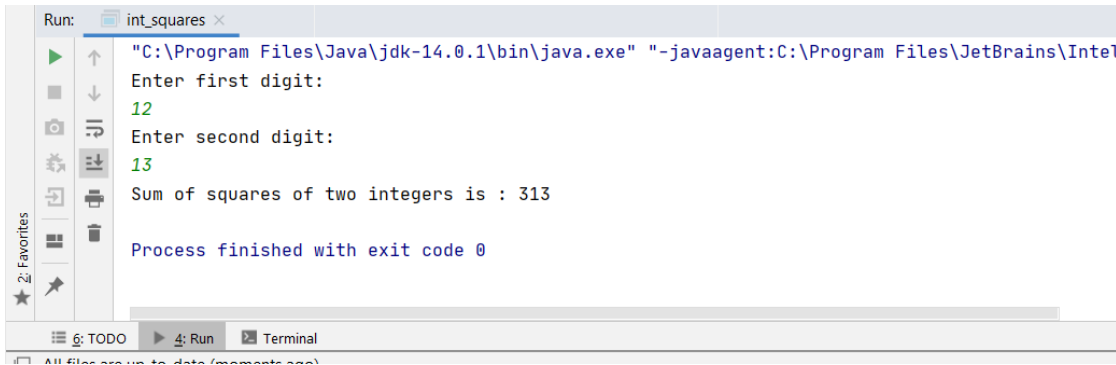
Output:


```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA C
Enter the number of elements to get sum:
5
1 2 3 4 5
The sum of your provided elements are: 15

Process finished with exit code 0
```

2.7 Consider any two integers. Write a program to print sum of their squares.

Code:

```java
import java.util.Scanner;
public class int_squares {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a,b,c,d,e;
        System.out.println("Enter first digit:");
        a=sc.nextInt();
        b = a*a;
        System.out.println("Enter second digit:");
        c = sc.nextInt();
        d = c*c;
        e = b+d;
        System.out.println("Sum of squares of two integers is : " + e);

    }
}
```
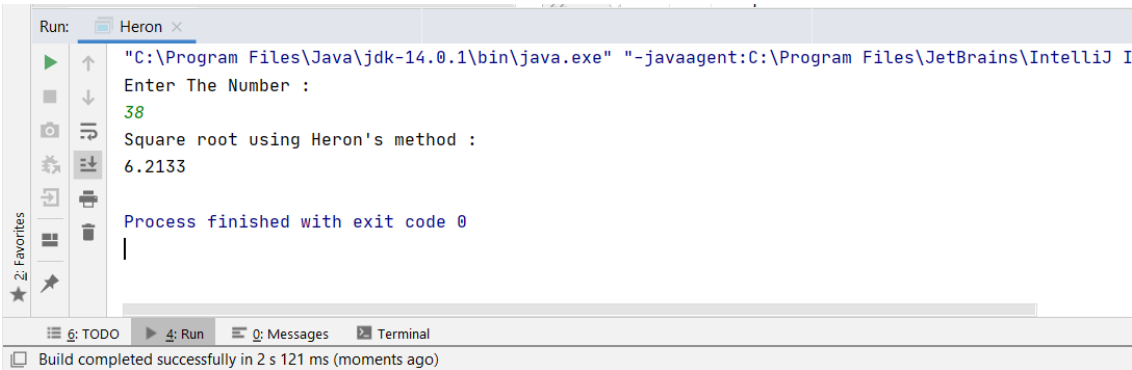
Output:


```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intel
Enter first digit:
12
Enter second digit:
13
Sum of squares of two integers is : 313

Process finished with exit code 0
```

2.8 Write a program to find square root of a given positive integer using Heron's method to find square root.

```java
import java.util.Scanner;
public class Heron {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter The Number  : ");
        int a = scanner.nextInt();
        System.out.println("Square root using heron's method : ");
        System.out.println((double)Math.round(heron(a) * 10000d) / 10000d);
    }
    public static int ClosetNumber(int a)  {
        int i, = a - 1;
        while (a != 0) {
            for (i = 1; i * i <= a; i++)
                if (i * i == a)
                    return a;
            a--;
        }
        return 0;
    }
    public static double heron(int x)
    {
        double a, i;
        a = ClosetNumber(x);
        for (i = 0; i < 4; i++)
            a = 0.5 * (a + x / a);
        return a;
    }
}
```



```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ I
Enter The Number :
38
Square root using Heron's method :
6.2133

Process finished with exit code 0
```
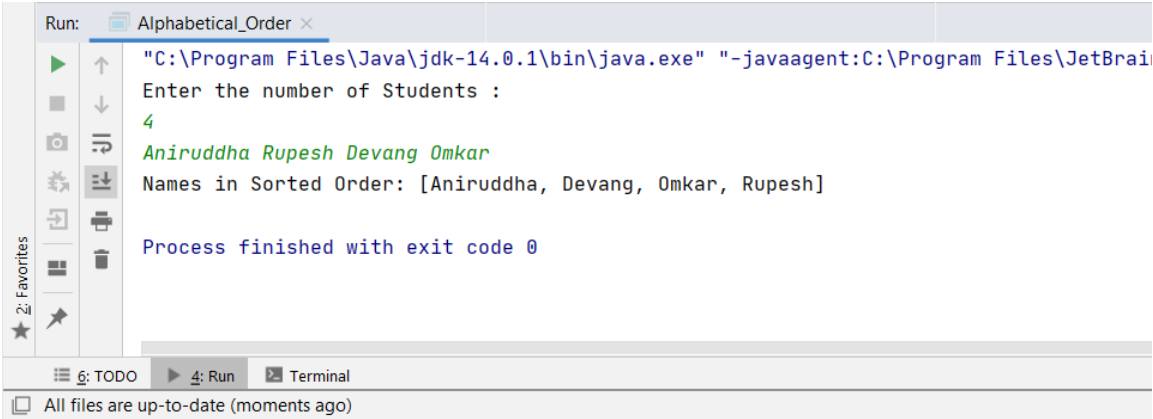
2.9 Write a program to sort and print the names of students taken from command line in alphabetical order.

Code :

```java
import java.util.Scanner;
public class Alphabetical_Order{
    public static void main(String[] args)  {
        int n;
        String temp;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number of names you want to enter:");
        n = s.nextInt();
        String names[] = new String[n];
        Scanner s1 = new Scanner(System.in);
        System.out.println("Enter all the names:");
        for(int i = 0; i < n; i++)
            names[i] = s1.nextLine();
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)       {
                if (names[i].compareTo(names[j])>0)     {
                    temp = names[i];
                    names[i] = names[j];
                    names[j] = temp;
                }
            }
        System.out.print("Names in Sorted Order:");
        for (int i = 0; i < n - 1; i++)
            System.out.print(names[i] + ",");
        System.out.print(names[n - 1]);
    }
}
```
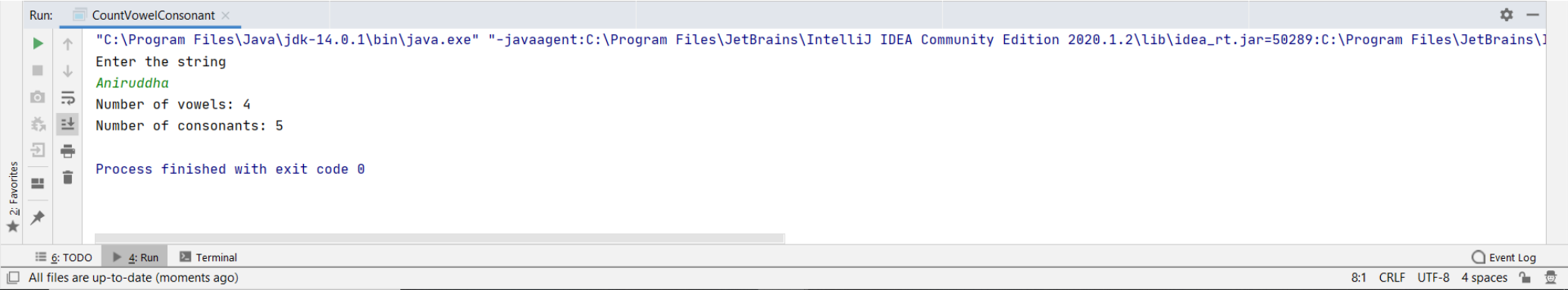
Output:


```
Run:    Alphabetical_Order ×
    "C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrai
    Enter the number of Students :
    4
    Aniruddha Rupesh Devang Omkar
    Names in Sorted Order: [Aniruddha, Devang, Omkar, Rupesh]

    Process finished with exit code 0
```

2.10 Write a program to print total numbers of vowels and consonants in a given string.

Code:

```java
import java.util.Scanner;
public class CountVowelConsonant {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int vCount = 0, cCount = 0;
        System.out.println("Enter the string ");
        String str = sc.nextLine();
        str = str.toLowerCase();
        for(int i = 0; i < str.length(); i++) {
            if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' || str.charAt(i) == 'u')
                vCount++;
            else if(str.charAt(i) >= 'a' && str.charAt(i)<='z')
                cCount++;
        }
        System.out.println("Number of vowels: " + vCount);
        System.out.println("Number of consonants: " + cCount);
    }
}
```

Output:


```
Run:    CountVowelConsonant ×
    "C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.2\lib\idea_rt.jar=50289:C:\Program Files\JetBrains\I
    Enter the string
    Aniruddha
    Number of vowels: 4
    Number of consonants: 5

    Process finished with exit code 0
```

2.11 Given two English words, write a program to check if the first word is anagram of the second word. (An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. (Example: Anagram of TOM MARVOLO RIDDLE is I AM LORD VOLDEMORT.)

Code:

```java
import java.util.Arrays;
import java.util.Scanner;
public class Anagram {
    static void areAnagram(String str1, String str2) {
```

```java
        String s1 = str1.replaceAll("\\s", "");
        String s2 = str2.replaceAll("\\s", "");

        boolean status = true;
        int n1 = s1.length();
        int n2 = s2.length();
        if (n1 != n2)
            status = false;
        char[] ArrayS1 = s1.toLowerCase().toCharArray();
        char[] ArrayS2 = s2.toLowerCase().toCharArray();
        Arrays.sort(ArrayS1);
        Arrays.sort(ArrayS2);
        status = Arrays.equals(ArrayS1, ArrayS2);
        if (status)
            System.out.println(str1 + " and " + str2 + " are anagrams");
        else
            System.out.println(str1 + " and " + str2 + " are not anagrams");
    }

    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter first string :");
        String str1 = in.nextLine();
        System.out.println("Enter second string :");
        String str2 = in.nextLine();
        areAnagram(str1, str2);
    }
}
```
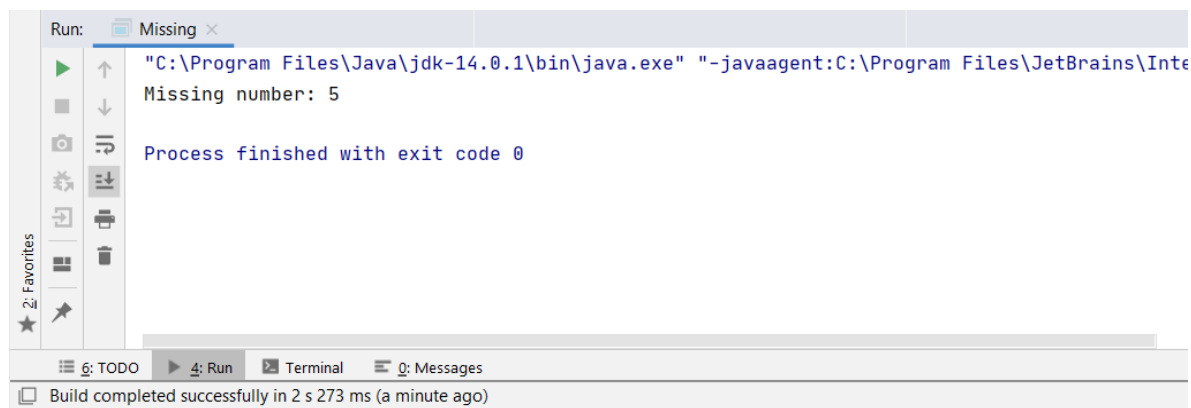
Output:

```
Run:      Anagram
    "C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
    Enter first string :
    Listen
    Enter second string :
    Silent
    Listen and Silent are anagrams

    Process finished with exit code 0
```
All files are up-to-date (a minute ago)

2.12 Write a program to print a missing number in a sorted integer array.

```java
public class Missing {
    static int search(int arr1[], int size){
        int a = 0, b = size - 1;
        int mid = 0;
        while ((b - a) > 1){
            mid = (a + b) / 2;
            if ((arr1[a] - a) != (arr1[mid] - mid))
                b = mid;
            else if ((arr1[b] - b) != (arr1[mid] - mid))
                a = mid;
        }
        return (arr1[mid] + 1);
    }
    public static void main (String[] args)  {
        int array[] = { 1, 2, 3, 4, 6, 7, 8, 9, 10 };
        int size = array.length;
        System.out.println("Missing number: " + search(array, size));
    }
}
```
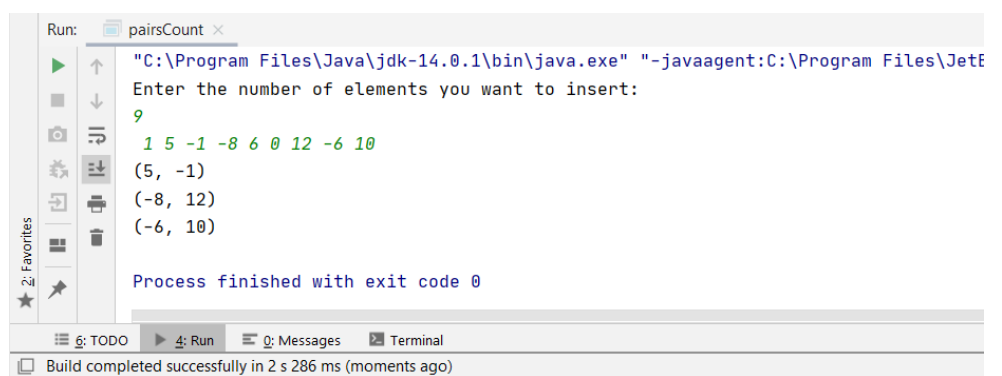
Output:

```
Run:      Missing
    "C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
    Missing number: 5

    Process finished with exit code 0
```
Build completed successfully in 2 s 273 ms (a minute ago)

2.13 Write a program to find all the pairs of numbers on an integer array whose sum is equal to a given number.

```java
import java.util.Scanner;
public class pairsCount {
    static void showPairs(int arr[], int n, int k) {
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)
                if (arr[i] + arr[j] == k)

                    System.out.println("(" + arr[i] + ", " + arr[j] + ")");
    }

    public static void main(String[] arg) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements
                you want to insert: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0; i<arr.length; i++)
            arr[i]=sc.nextInt();

        int k = 4;
        showPairs(arr, n, k);
    }
}
```

Output:

```
Run:      pairsCount
    "C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetB
    Enter the number of elements you want to insert:
    9
    1 5 -1 -8 6 0 12 -6 10
    (5, -1)
    (-8, 12)
    (-6, 10)

    Process finished with exit code 0
```
Build completed successfully in 2 s 286 ms (moments ago)

Conclusion: We understood and performed various programs using Java.