

## Experiment: 1

**Aim:** Write a Program to accept a String as a command-line argument and print a Welcome message as given below:- “Welcome your name”.

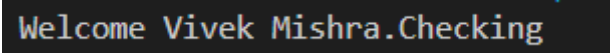
**Theory:**

To create a program that accepts a String as a command-line argument and prints a welcome message, we can follow these steps in Java: first, accept the command-line argument using the args array; second, manipulate the string by concatenating the name with the welcome message; and finally, print the welcome message to the console using the System.out.println() method. This will allow us to create a simple and effective program that can accept user input and provide a personalized welcome message as output

**Code:**

```
public class pr1 {  
    public static void main(String[] args) {  
        String s="Checking";  
        System.out.println("Welcome Vivek Mishra."+s);  
    }  
}
```

**Output:**

A screenshot of a terminal window showing the output of the Java program. The text "Welcome Vivek Mishra.Checking" is displayed in a monospaced font on a dark background.

Welcome Vivek Mishra.Checking

## Experiment: 2

**Aim:** Program to find ASCII code of a character.

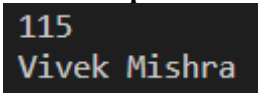
**Theory:**

**ASCII Code:** ASCII (American Standard Code for Information Interchange) is a character encoding standard used in computers and other devices. Each character in the ASCII standard is represented by a unique code, which is a numeric value between 0 and 127. **Character data type:** In Java, the char data type is used to represent a single character. Each char variable occupies 2 bytes of memory and can hold a Unicode character code. **Type casting:** To find the ASCII code of a character, we can use type casting to convert the char variable to an int variable. When a char is cast to an int, its Unicode value is assigned to the int variable. Since the ASCII codes are a subset of the Unicode character set, the value of the int variable will be the ASCII code of the character.

**Code:**

```
public class pr2 {  
    public static void main(String[] args) {  
        char c1='s';  
        int x=c1;  
        System.out.println(x);  
        System.out.println("Vivek Mishra");  
    }  
}
```

**Output:**



```
115  
Vivek Mishra
```

### Experiment: 3

**Aim:** Write a Program to accept two integers as inputs and print their sum.

**Theory:**

To create a program that accepts two integers as inputs and prints their sum, we can use the Scanner class in Java to accept user input. Once we have the input integers, we can add them together using the + operator in Java. Finally, we can print the sum to the console or display it to the user using the System.out.println() method.

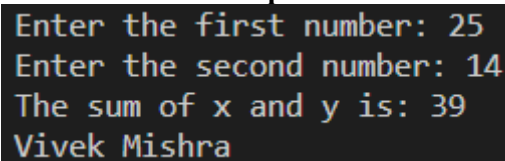
By following these steps, we can create a simple and effective program that can accept user input and provide the sum of two integers as output.

**Code:**

```
import java.util.Scanner;

public class pr3 {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the first number: ");
        int x=s.nextInt();
        System.out.print("Enter the second number: ");
        int y=s.nextInt();
        System.out.println("The sum of x and y is: "+(x+y));
        System.out.println("Vivek Mishra");
    }
}
```

**Output:**

A screenshot of a terminal window showing the output of the Java program. The text is as follows:

```
Enter the first number: 25
Enter the second number: 14
The sum of x and y is: 39
Vivek Mishra
```

### Experiment: 4

**Aim:** Swapping two numbers using bitwise operator.

**Theory:**

1. XOR operator: The XOR (^) operator can be used to swap two numbers without using a temporary variable. XOR operator returns a 1 in each bit position where the corresponding bits of either operand are 1, but not both.
2. Swapping: To swap two numbers using the XOR operator, we can perform the following steps:
  - a. Assign the first number to a temporary variable.
  - b. XOR the first number with the second number, and store the result in the first number.
  - c. XOR the first number with the temporary variable, and store the result in the second number.

**Code:**

```
import java.util.Scanner;

public class pr4 {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the first number: ");
        int x=s.nextInt();
        System.out.print("Enter the second number: ");
        int y=s.nextInt();
        System.out.println("Value of x is: "+x+" Value of y is: "+y);
        x=x^y;
        y=x^y;
        x=x^y;
        System.out.println("Swaped: Value of x is: "+x+" Value of y is:"+y);
        System.out.println("Vivek Mishra");
    }
}
```

**Output:**

```
Enter the first number: 15
Enter the second number: 25
Value of x is: 15 Value of y is: 25
Swaped: Value of x is: 25 Value of y is:15
Vivek Mishra
```

### Experiment: 5

**Aim:** Initialize two-character variables in a program and display the characters in alphabetical order.

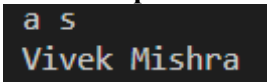
#### Theory:

1. Variable initialization: To initialize two-character variables in a program, we can declare two char variables and assign values to them. In this experiment, we will initialize the two variables with arbitrary characters.
2. Sorting: To display the characters in alphabetical order, we need to compare the values of the two variables and sort them in ascending order. We can use the if-else statement to compare the values of the two variables and swap them if necessary.
3. Testing: After writing the program, we can test it with different sets of characters to ensure that it correctly sorts the variables in alphabetical order.

#### Code:

```
public class pr5 {  
    public static void main(String[] args) {  
  
        char ch1='s';  
        char ch2='a';  
        if(ch1<ch2){  
            System.out.println(ch1+" "+ch2);  
        }  
        else{  
            System.out.println(ch2+" "+ch1);  
        }  
        System.out.println("Vivek Mishra");  
    }  
}
```

#### Output:



```
a s  
Vivek Mishra
```

## Experiment: 6

**Aim:** Write a program to receive a colour code from the user (an Alphabet).

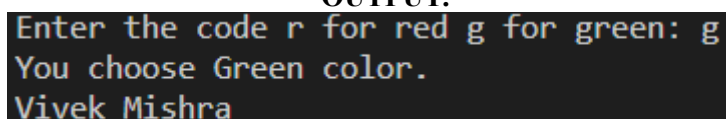
### Theory:

**Data types:** In Java, the char data type is used to represent a single character. Since we only need to read a single alphabet (color code) from the user, we can use the char data type to store the input. **User input:** To accept input from the user, we can use the Scanner class in Java. The Scanner class allows us to read input from different sources, such as the keyboard or a file. In this experiment, we will read input from the keyboard using the Scanner class. **Reading input:** To read input from the user, we can use the next() method of the Scanner class. This method reads the next input string entered by the user and returns it as a String data type. **Extracting the first character:** Since we only need to read a single character from the input string, we can use the charAt() method to extract the first character from the input string and store it as a char data type.

### CODE:

```
public class pr6 {  
    public static void main(String[] args) {  
        System.out.print("Enter the code r for red g for green: ");  
        Scanner s = new Scanner(System.in);  
        char x = s.next().charAt(0);  
        if(x=='r'){  
            System.out.println("You choose Red color.");  
        }  
        else if(x=='g'){  
            System.out.println("You choose Green color.");  
        }  
        System.out.println("Vivek Mishra");  
    }  
}
```

### OUTPUT:

A screenshot of a terminal window showing the output of the Java program. The text is displayed on a dark background with light-colored characters. The output consists of three lines: a prompt for input, a confirmation message, and the program's name.

```
Enter the code r for red g for green: g  
You choose Green color.  
Vivek Mishra
```

### Experiment: 7

**Aim:** Write a program to print even numbers between 23 and 57. Each number should be printed in a separate row.

#### Theory:

In this program, we first initialize the starting and ending numbers (23 and 57) using integer variables. Then we use a for loop to iterate through each number between the starting and ending values. Inside the loop, we use an if statement to check if the current number is even by using the modulus operator % to check if there is a remainder when dividing by 2. If the current number is even, we print it to the console using the System.out.println() method. This results in each even number being printed on a separate line.

#### Code:

```
import java.util.*;
public class Experiment_7 {
    public static void main(String[] args) {
        int start = 23;
        int end = 57;
        for (int i = start; i <= end; i++) {
            if (i % 2 == 0) {
                System.out.println(i+" ");
            }
        }
        System.out.println("Vivek Mishra");
    }
}
```

#### Output:

```
24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
Vivek Mishra
```

### Experiment: 8

**Aim:** Write a program to print \* in Floyd's format (using for and while loop).

**Theory:**

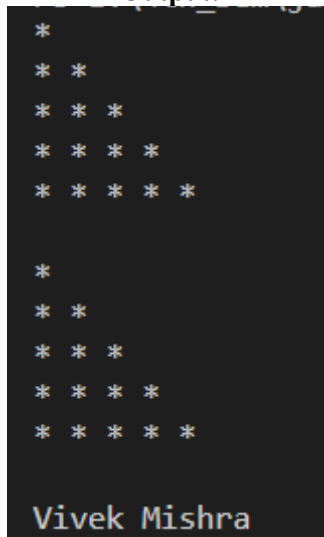
The Floyd's triangle is a right-angled triangular array of numbers or symbols, where the first row has one element, the second row has two elements, and so on. The triangle is named after Robert Floyd, who was an American computer scientist. To print the Floyd's triangle in asterisks, we can use either a for loop or a while loop. In both cases, we initialize a count variable to keep track of the number of asterisks printed. We then use nested loops to iterate through each row and column of the triangle. In the for loop version, we use a nested for loop to iterate through each row and column of the triangle. We use the outer loop to iterate through each row, and the inner loop to print the correct number of asterisks for that row.

Source code:

```
import java.util.*;
public class pr8 {
    public static void main(String[] args) {
        int r = 5;
        for (int i = 1; i <= r; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
        int x=0;

        int x1=0;
        while (x<=5) {
            int y=0;
            while(y<x){
                System.out.print("* ");
                y++;
            }
            System.out.println();
            x++;
        }
    }
    System.out.println("Vivek Mishra");
}
```

**Output:**



```
*
* *
* * *
* * * *
* * * * *

*
* *
* * *
* * * *
* * * * *
```

Vivek Mishra



## Experiment-9

**Aim:** Write a Java program to find if the given number is palindrome or not.

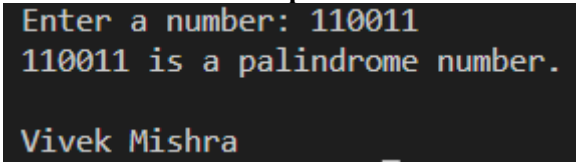
### Theory:

In this program, we first read the input from the user using the Scanner class. We then initialize a temporary variable to store the input number and a variable to store the reversed number. Using a while loop, we reverse the input number by extracting each digit and adding it to the reversed.

### Code:

```
import java.util.*;
public class Experiment_5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        sc.close();
        int temp = num;
        int rev = 0;
        while (temp > 0) {
            int digit = temp % 10;
            rev = rev * 10 + digit;
            temp /= 10;
        }
        if (num == rev) {
            System.out.println(num + " is a palindrome number.");
        } else {
            System.out.println(num + " is not a palindrome number.");
        }
    }
}
```

### Output:

A screenshot of a terminal window showing the output of the Java program. The text is as follows:  
Enter a number: 110011  
110011 is a palindrome number.  
Vivek Mishra

### Experiment:10

**Aim:** Initialize an integer array with ASCII values and print the corresponding character values in a single row.

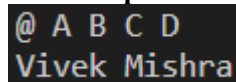
**Theory:**

In this program, we first declare and initialize an integer array `ascii Values` with some ASCII values. We then use a for loop to iterate over the array. For each element in the array, we convert the ASCII value to its corresponding character using a typecast to `char`. Finally, we print the character value in a single row separated by spaces using the `System.out.print()` method.

**Code:**

```
public class pr10 {  
    public static void main(String[] args) {  
        int [] asiiVal={64,65,66,67,68};  
        for (int i = 0; i < asiiVal.length; i++) {  
            System.out.print((char)asiiVal[i]+ " ");  
        }  
        System.out.println("\nVivek Mishra");  
    }  
}
```

**Output:**



```
@ A B C D  
Vivek Mishra
```

### Experiment:11

**Aim:** Write a program to reverse the elements of a given 2\*2 array. Four integer numbers need to be passed as Command-Line arguments.

**Theory:** Take input the size of the array and the elements of the array. Consider a function reverse which takes the parameters-the array(say arr) and the size of the array(say n). Inside the function, a new array (with the array size of the first array, arr) is initialized. The array arr[] is iterated from the first element, and each element of array arr[] is placed in the new array from the back, i.e., the new array is iterated from its last element.

In this way, all the elements of the array arr[] are placed reversely in the new array. Further, we can iterate through the new array from the beginning and print the elements of the array.

**Code:**

```
import java.util.Scanner;
public class pr11 {
    public static int[][] revers(int [][]arr){
        int arr1 [][] = new int[2][2];
        for(int i=0; i<2; i++){
            for (int j = 0; j < arr.length; j++) {
                arr1[i][j]=arr[j][i];
            }
        }
        return arr1;
    }
    public static void disp(int [][] arr){
        for(int i=0; i<arr.length; i++){
            for (int j = 0; j < arr.length; j++) {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println("");
        }
    }
    public static void main(String[] args) {
        int arr [][] = new int[2][2];
        Scanner s= new Scanner(System.in);
        System.out.print("Enter the element of arr: ");
        for(int i=0; i<2; i++){
            for (int j = 0; j < arr.length; j++) {
                arr[i][j]=s.nextInt();
            }
        }
        int arrres[][]=revers(arr);
        System.out.println("Give matrix: ");
        disp(arr);
        System.out.println("Resultant matrix: ");
        disp(arrres);
        System.out.println("Vivek Mishra");
    }
}
```

**Output:**

```
Enter the element of arr: 1 2 3 4
Give matrix:
1 2
3 4
Resultant matrix:
1 3
2 4
Vivek Mishra
```

## Experiment:12

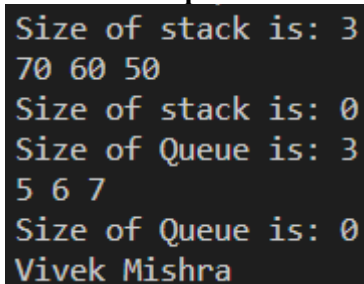
**Aim:** Create a java program to implement stack and queue concept.

**Theory:** The stack is a linear data structure that is used to store the collection of objects. It is based on Last-In-First-Out (LIFO). **Java collection** framework provides many interfaces and classes to store the collection of objects. One of them is the Stack class that provides different operations such as push, pop, search, etc. The Queue interface is present in [java.util](#) package and extends the [Collection interface](#) is used to hold the elements about to be processed in FIFO(First In First Out) order. It is an ordered list of objects with its use limited to inserting elements at the end of the list and deleting elements from the start of the list, (i.e.), it follows the FIFO or the First-In-First-Out principle.

**Code:**

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;
public class pr12 {
    public static void main(String[] args) {
        Stack<Integer> st = new Stack<>();
        st.push(50);
        st.push(60);
        st.push(70);
        System.out.println("Size of stack is: "+st.size());
        for (int i = st.size()-1; i >=0 ; i--) {
            int x=st.peek();
            System.out.print(x+ " ");
            st.pop();
        }
        System.out.println("\nSize of stack is: "+st.size());
        Queue<Integer> q1 = new LinkedList();
        q1.add(5);
        q1.add(6);
        q1.add(7);
        System.out.println("Size of Queue is: "+q1.size());
        for (int i = q1.size()-1; i >=0 ; i--) {
            int x1=q1.peek();
            System.out.print(x1+ " ");
            q1.remove();
        }
        System.out.println("\nSize of Queue is: "+q1.size());
        System.out.println("Vivek Mishra");
    }
}
```

**Output:**



```
Size of stack is: 3
70 60 50
Size of stack is: 0
Size of Queue is: 3
5 6 7
Size of Queue is: 0
Vivek Mishra
```

### Experiment:13

**Aim:** Write a java program to produce the tokens from given long string.

**Theory:** The Java compiler breaks the line of code into text (words) is called Java tokens. These are the smallest element of the **Java program**. The Java compiler identified these words as tokens. These tokens are separated by the delimiters. It is useful for compilers to detect errors. Remember that the delimiters are not part of the Java tokens.

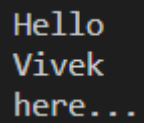
**Code:**

```
import java.util.*;
public class pr13 {
    public static void main(String args[])
    {

        StringTokenizer st1 = new StringTokenizer(
            "Hello Vivek here...", " ");
        while (st1.hasMoreTokens()){
            System.out.println(st1.nextToken());
        }

    }
}
```

**Output:**



```
Hello
Vivek
here...
```

### Experiment:14

**Aim:** Using the concept of method overloading Write method for calculating the area of triangle, circle and rectangle.

**Theory:** Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

**Code:**

```
import java.util.Scanner;

public class pr14 {
    public static double area(double l , double b, double h){
        double x= l*b*h;
        return x;
    }
    public static double area(double base , double hight ){
        double x = 0.5*base*hight;
        return x;
    }
    public static double area( double r){
        double x = 2*3.14*r;
        return x;
    }
    public static void main(String[] args) {
        System.out.print("Enter the length, branth and hight of Rectangle:");
        Scanner x = new Scanner(System.in);
        double len= x.nextDouble();
        double branth= x.nextDouble();
        double hight= x.nextDouble();
        System.out.print("Enter the base and hight of tringle:");
        double base =x.nextDouble();
        double hightT= x.nextDouble();
        System.out.print("Enter the Radius of circle: ");
        double r=x.nextDouble();

        System.out.println("Area of Rectagle is: "+area(len,branth, hight));
        System.out.println("Area of Triangle is: "+area(base, hightT));
        System.out.println("Area of Circle is: "+area(r));
        System.out.println("Vivek Mishra");

    }
}
```

**Output:**

```
Enter the length, branth and hight of Rectangle:5.4 2 3.3
Enter the base and hight of tringle:2.3 5.6
Enter the Radius of circle: 4.5
Area of Rectagle is: 35.64
Area of Triangle is: 6.4399999999999995
Area of Circle is: 28.26
Vivek Mishra
```