***JAVA PROGRAMMING LAB***

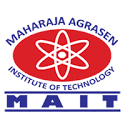
***ETCS - 357***

DR. ASHISH KHANNA MADHAV KHURANA

ROLL NO.: 05914802718

SEMESTER - 5TH

BATCH: C4



Maharaja Agrasen Institute of Technology, PSP Area, Sector – 22, Rohini, New Delhi – 110085

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **NAME OF EXPERIMENT** | **DATE** | **SIGNATURE** |
| 1. | Write a program to check Vowel or consonant using switch case. | 26/08/20 |  |
| 2. | Write a java program to print prime numbers first n prime numbers. | 26/08/20 |  |
| 3. | Write a java program to check whether the input year is leap or not. | 26/08/20 |  |
| 4. | Write an application that accepts two doubles as its command line arguments, multiple these together and display the product. | 26/08/20 |  |
| 5. | Write an application that accepts radius of a circle as its command line argument display the area. | 26/08/20 |  |
| 6. | Write a program to calculate the sum of elements before and after the given index in an Array. | 02/09/20 |  |
| 7. | Write a Program that creates an Object of Class Person. It should set the values of instance variables and then display these values as Output on command line. | 02/09/20 |  |
| 8. | Create a java program to implement the concept of Stack without using Stack Class. | 02/09/20 |  |
| 9. | Create a java program to display ASCII value of a character | 02/09/20 |  |
| 10. | Create a java program to convert character to string | 02/09/20 |  |
| 11. | Create a java program to check whether the given number is palindrome or not | 02/09/20 |  |
| 12. | Create a java program to implement stack and queue concept | 02/09/20 |  |
| 13. | Create a java calculate the factorial of a number | 07/09/20 |  |
| 14. | WAP to find array index, where preceding and succeeding sum to it are equal. | 07/09/20 |  |
| 15. | WAP to initialize four element array and calculate its avg. | 07/09/20 |  |
| 16. | WAP to sort an array | 07/09/20 |  |
| 17. | Using method overloading calculate area of triangle, circle and rectangle. | 07/09/20 |  |
| 18. | WAP to create all possible permutations of distinct integers. | 07/09/20 |  |
| 19. | WAP to take input as number and output it as word. | 07/09/20 |  |
| 20. | WAP to check whether string is palindrome or not. | 07/09/20 |  |
| 21. | WAP to divide string into n equal parts. | 07/09/20 |  |
| 22. | WAP to count number of instances created. | 07/09/20 |  |
| 23. | Write a class and initialize its data members. | 23/09/20 |  |
| 24. | Write a circle and initialize its data members using constructor and implement constructor overloading | 23/09/20 |  |
| 25. | WAP to create class and overload its constructors and display all details about circle. | 23/09/20 |  |
| 26. | WAP to define shape class of abstract object. It should also have area function | 23/09/20 |  |
| 27. | Derive two sub-classes using a parent class and perform operations. | 23/09/20 |  |
| 28. | Create class person which contains all information. Derive class employee using it. | 23/09/20 |  |
| 29. | Implement author class having only private members. Implement book class using it. | 23/09/20 |  |
| 30. | Implement Circle and Cylinder class. | 07/10/20 |  |
| 31. | WAP to implement simple inheritance. | 07/10/20 |  |
| 32. | WAP to implement method overriding. | 07/10/20 |  |
| 33. | WAP to implement multiple Inheritance. | 07/10/20 |  |
| 34. | WAP to implement super keyword. | 07/10/20 |  |
| 35. | WAP to illustrate a interface and implement it. | 18/11/20 |  |
| 36. | WAP to implement a interface using single inheritance | 18/11/20 |  |
| 37. | WAP to implement interface using multilevel inheritance | 18/11/20 |  |
| 38. | WAP to execute two threads, one displays every 1000ms and other at every 3000ms. Do it by extending thread class. | 18/11/20 |  |
| 39. | WAP to implement multi threaded program. First thread generates random number every second. Second thread calculates its square if even and third calculates its cube if odd. | 18/11/20 |  |

**EXPERIMENT NO: 1**

**AIM: Write a program to check Vowel or consonant using switch case.**

**SOURCE CODE:**

import java.util.Scanner;

class JavaExample

{

public static void main(String[ ] arg)

{

boolean isVowel=false;;

Scanner scanner=new Scanner(System.in);

System.out.println("Enter a character : ");

char ch=scanner.next().charAt(0);

scanner.close();

switch(ch)

{

case 'a' :

case 'e' :

case 'i' :

case 'o' :

case 'u' :

case 'A' :

case 'E' :

case 'I' :

case 'O' :

case 'U' : isVowel = true;

}

if(isVowel == true) {

System.out.println(ch+" is a Vowel");

}

else {

if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z'))

System.out.println(ch+" is a Consonant");

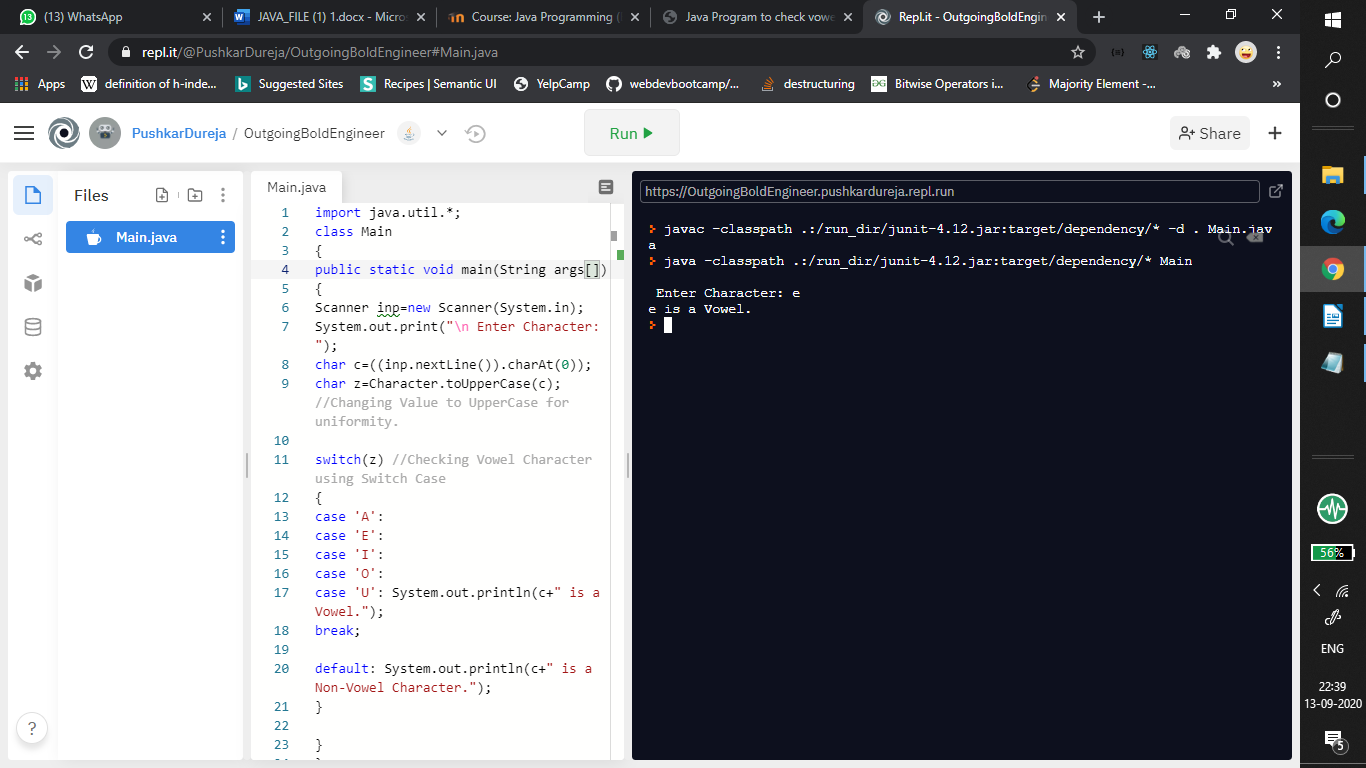
else

System.out.println("Input is not an alphabet");

}

}

}



**EXPERIMENT NO: 2**

**AIM:**

**Write a java program to print prime numbers within a range**

**CODE:**

import java.util.Scanner;

class primes

{static void primes(int N)

{

int i, j, flag;

for (i = 1; i <= N; i++)

{

if (i == 1 || i == 0)

continue;

flag = 1;

for (j = 2; j <= i / 2; ++j)

{

if (i % j == 0)

{

flag = 0;

break;

}}

if (flag == 1)

System.out.print(i + " ");

}}

public static void main (String[] args)

{

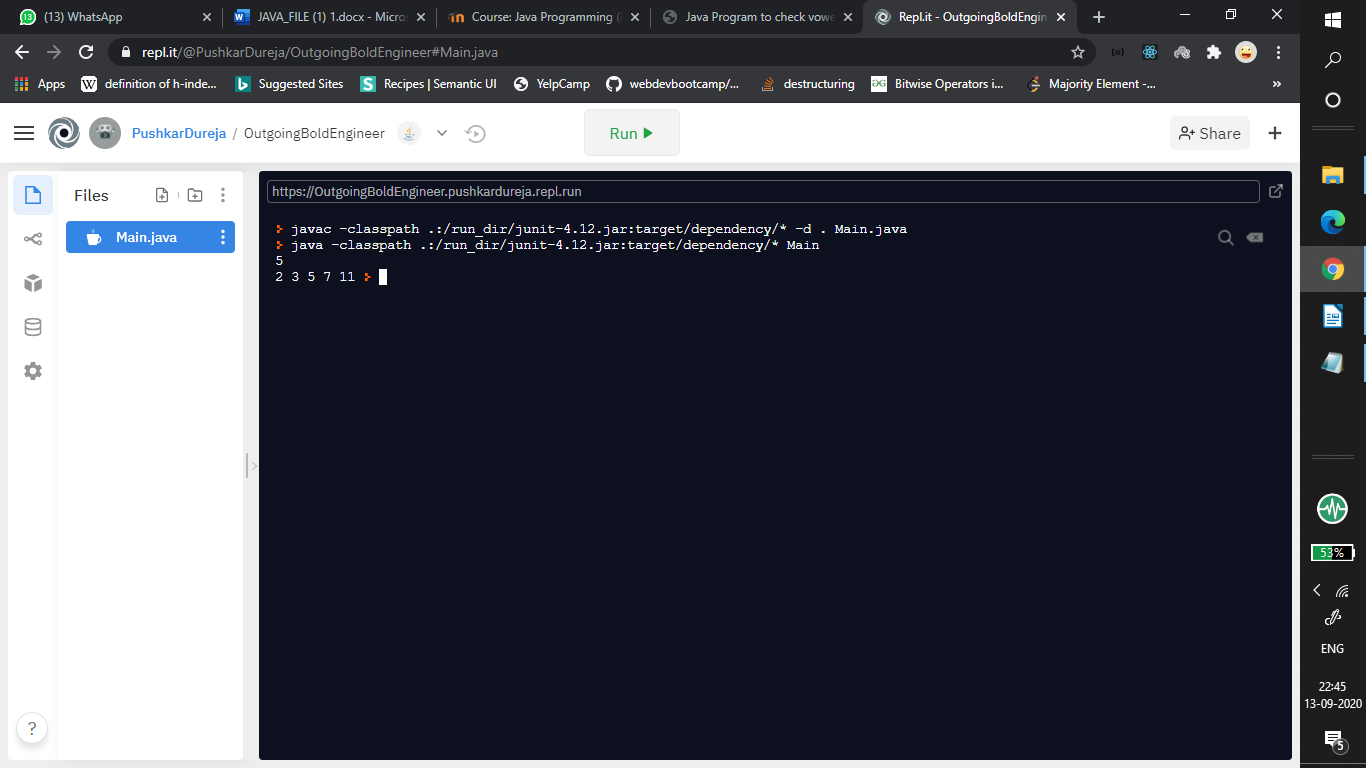
Scanner reader = new Scanner(System.in);

System.out.println("Enter n");

int N=reader.nextInt();

primes(N);

}}



**EXPERIMENT NO: 3**

**AIM: Write a Program to check whether the input year is leap or not**

**SOURCE CODE:**

import java.util.\*;

import java.lang.\*;

public class Main

{

public static void main(String[] args)

{

System.out.println("Input the year you want to check");

Scanner obj = new Scanner(System.in);

int year = obj.nextInt();

if (year%4 == 0 && year%100 != 0) {

System.out.println("Given year " +year+" is a leap year");

return;

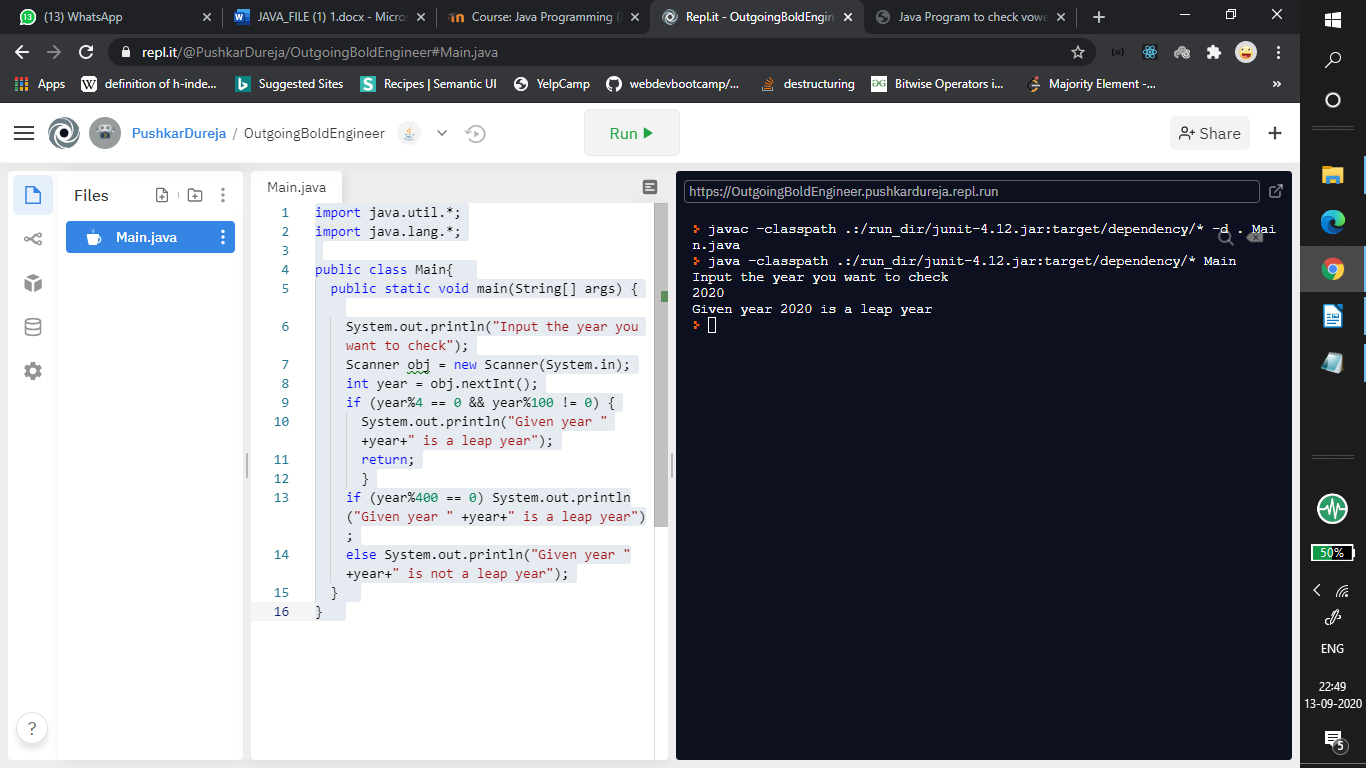
}

if (year%400 == 0) System.out.println("Given year " +year+" is a leap year");

else System.out.println("Given year "+year+" is not a leap year");

}

}



**EXPERIMENT NO: 4**

**AIM: Write an application that accepts two doubles as its command line arguments, multiple these together and display the product.**

**SOURCE CODE:**

import java.util.\*;

import java.lang.\*;

public class Main{

public static void main(String[] args)

{

int product = 1;

for(String str: args) {

int argument = Integer.parseInt(str);

product \*= argument;

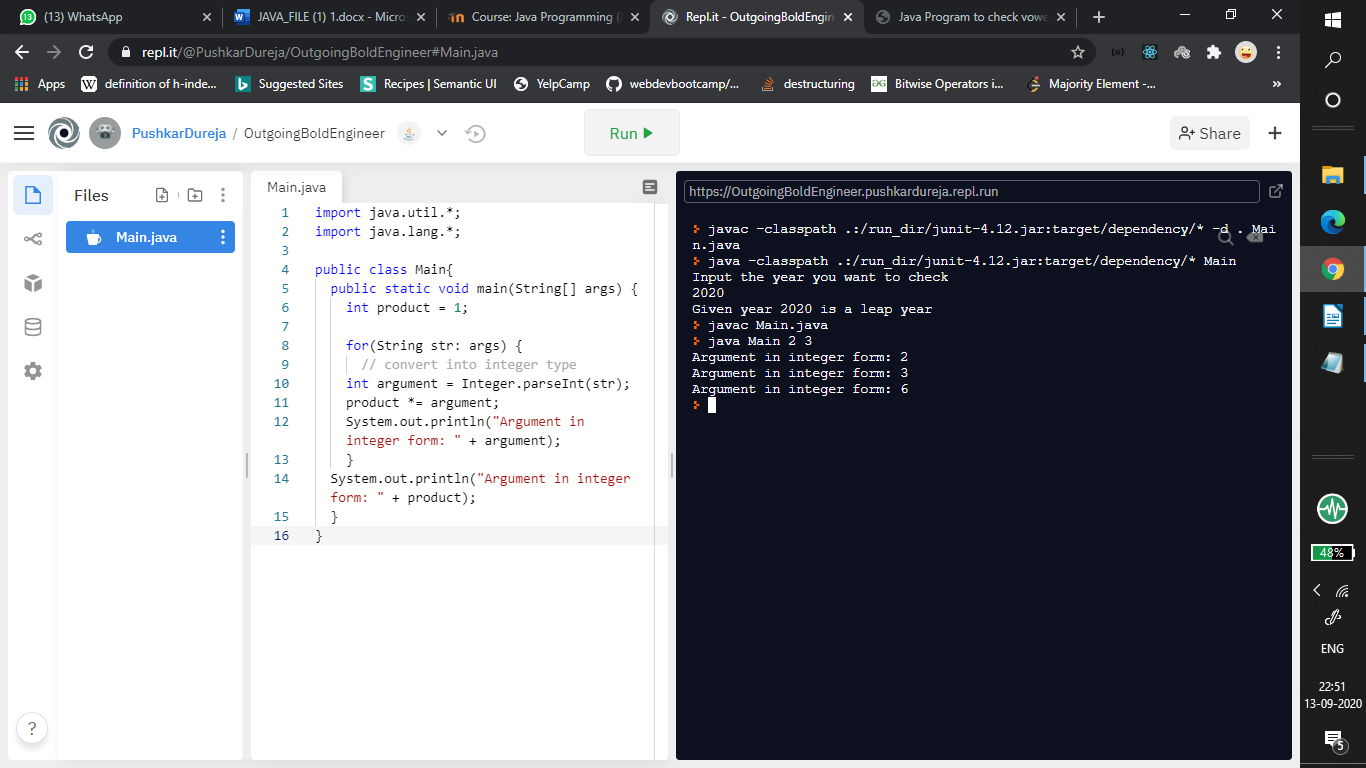
System.out.println("Argument in integer form: " + argument);

}

System.out.println("Argument in integer form: " + product);

}

}



**EXPERIMENT NO: 5**

**AIM: Write an application that accepts radius of a circle as its command line argument display the area.**

**SOURCE CODE:**

import java.util.\*;

import java.lang.\*;

public class Main{

public static void main(String[] args) {

double area;

for(String str: args)

{

double argument = Integer.parseInt(str);

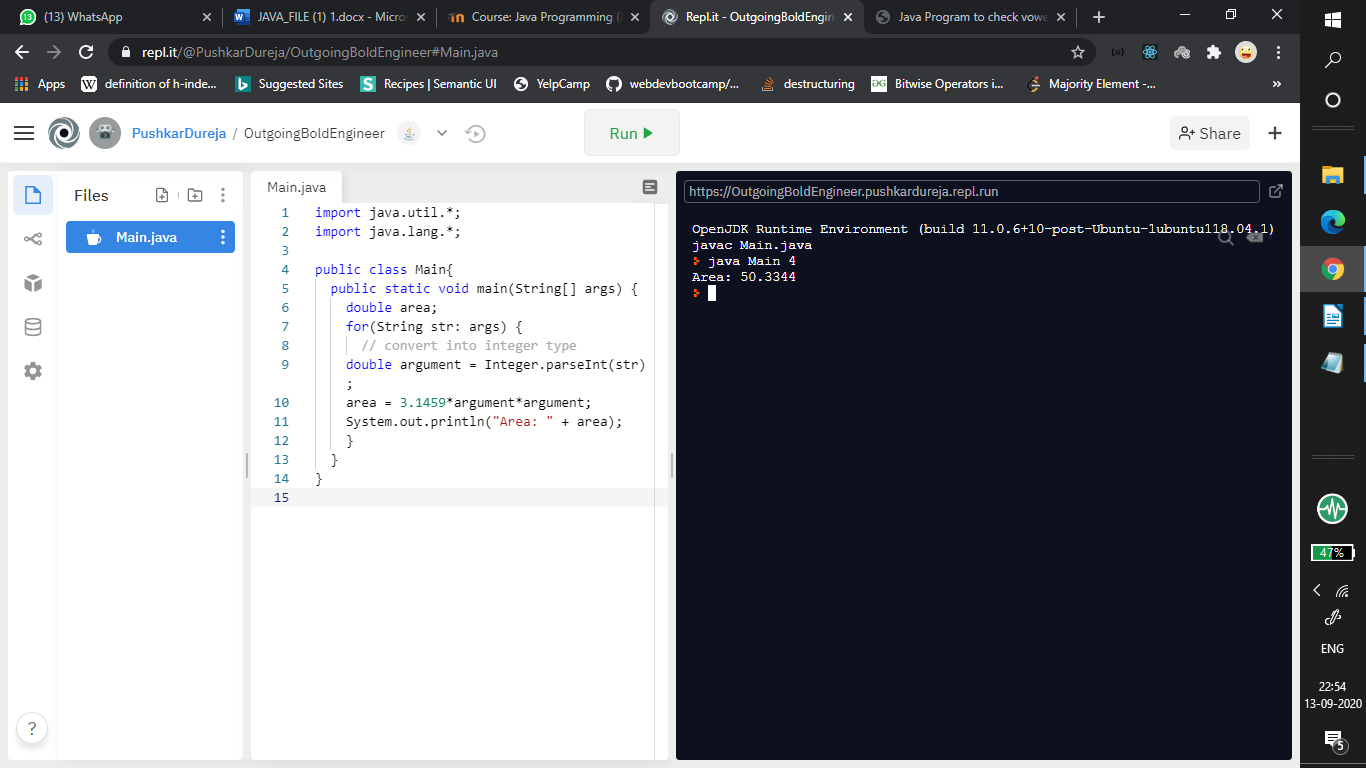
area = 3.1459\*argument\*argument;

System.out.println("Area: " + area);

}

}

}



**EXPERIMENT NO: 6**

**AIM: Write a program to calculate the sum of elements before and after the given index in an Array. Eg. If 4 is passed, the program should calculate sum of 4 numbers from beginning and sum of remaining numbers separately.**

**SOURCE CODE:**

import java.util.\*;

import java.lang.\*;

public class Main

{

public static void main(String[] args) {

int arr[] = {1,2,3,4,5,6};

int sum = 0;

Scanner obj = new Scanner(System.in);

System.out.println("Enter Index");

int idx = obj.nextInt();

if(idx>arr.length || idx <0)

System.out.println("Out of Bounds");

else

{

for(int i = 0;i<arr.length;i++){

if(i!=idx){

sum = sum + arr[i];

}

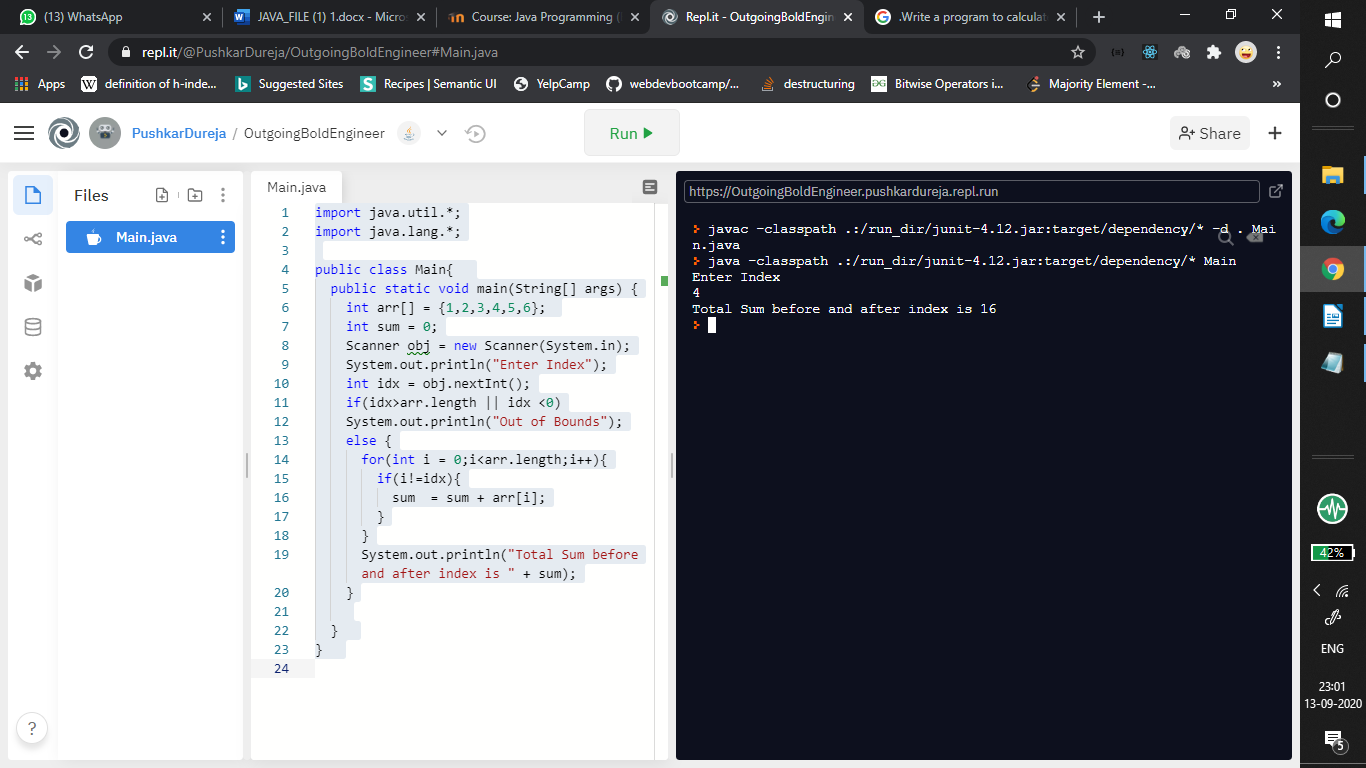
}

System.out.println("Total Sum before and after index is " + sum);

}

}

}



**EXPERIMENT NO: 7**

**AIM: Write a Program that describes a class Person and creates an Object of Class Person. The class should have instance variables to record name, age and salary. It should set the values of instance variables and then display these values as Output on command line.**

**SOURCE CODE:**

import java.util.\*;

import java.lang.\*;

class Person{

String name;

int age;

long salary;

Person(String n,int a,long sal){

name = n;

age = a;

salary = sal;

}

}

class Main {

public static void main(String args[]){

Person p = new Person("Khushi",19,50000);

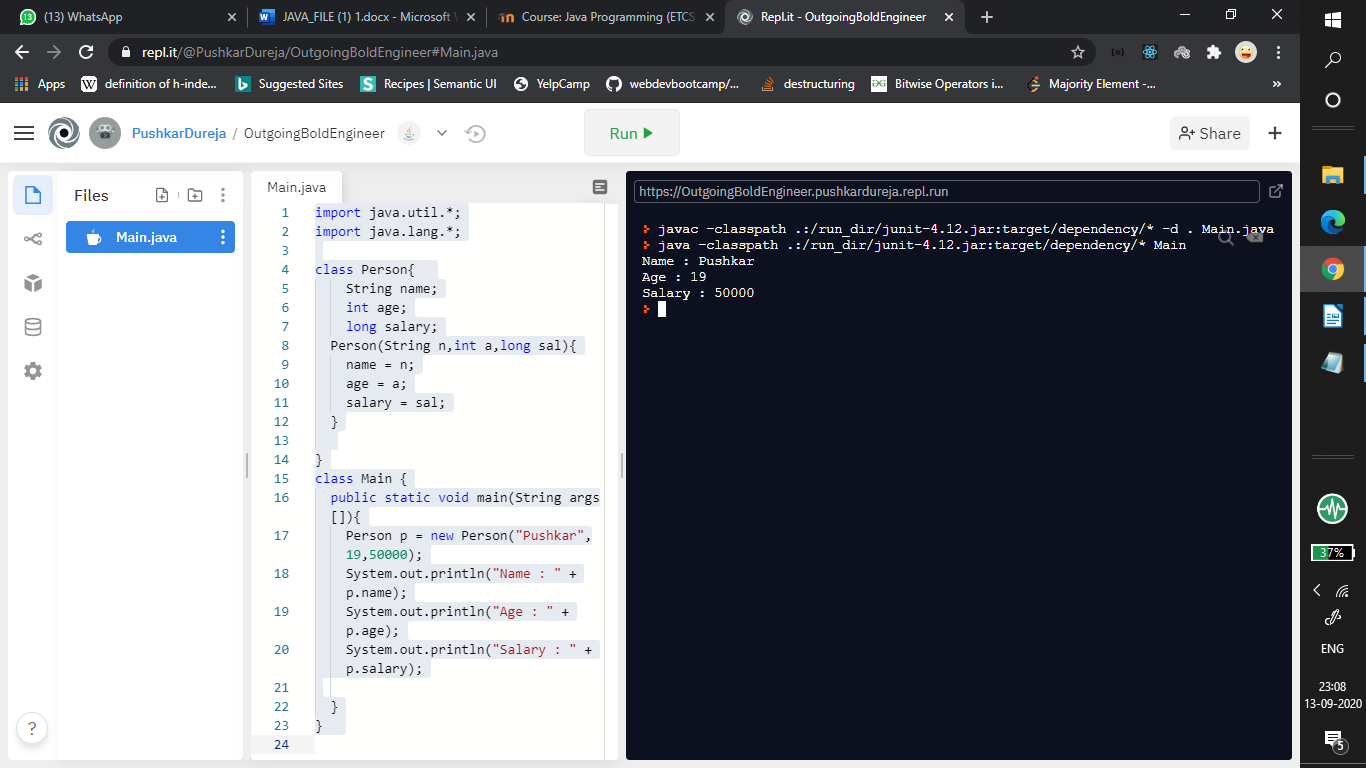
System.out.println("Name : " + p.name);

System.out.println("Age : " + p.age);

System.out.println("Salary : " + p.salary);

}

}



**EXPERIMENT NO: 8**

**AIM: Create a java program to implement the concept of Stack without using Stack Class.**

**SOURCE CODE:**

import java.util.ArrayList;

import java.util.Scanner;

class Main {

public static void display( ArrayList<Integer> arr){

for(int i = 0;i<arr.size();i++){

System.out.println(arr.get(i));

}

}

public static void main(String[] args) {

ArrayList<Integer> arr = new ArrayList<Integer>();

Scanner obj = new Scanner(System.in);

int ans = 1;

do{

System.out.println("2. Push");

System.out.println("3. Pop");

int input = obj.nextInt();

if(input ==2){

System.out.println("enter Number to be added to stack");

int n = obj.nextInt();

arr.add(n);

System.out.println("Stack :-");

display(arr);

}

else if(input == 3){

if(arr.size()!=0){

arr.remove(arr.size()-1);

System.out.println("Stack :-");

display(arr);

}

else {

System.out.println("stack is empty");

}

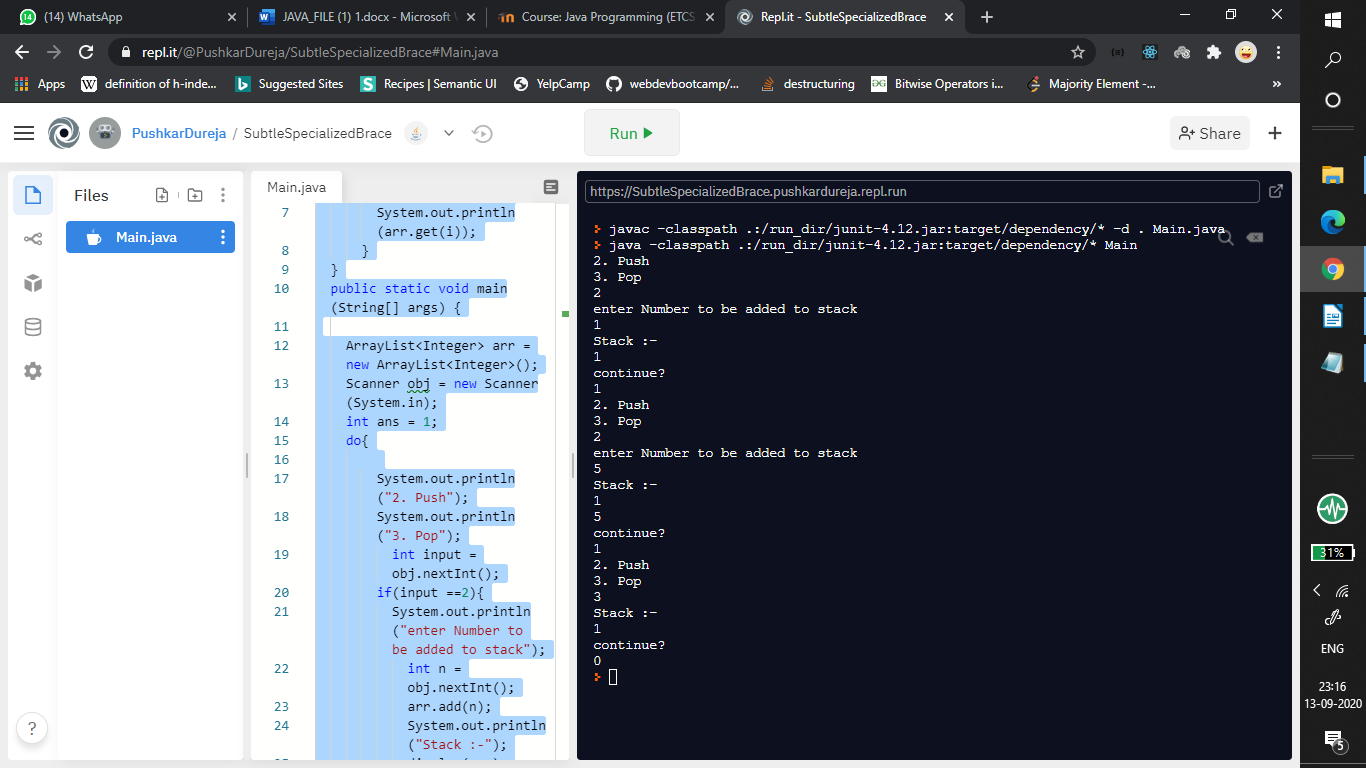
}

System.out.println("continue?");

ans = obj.nextInt();

}while(ans == 1);

}}



**EXPERIMENT NO: 9**

**AIM:**

**Write a java program to find the ASCII code of the entered character**

**CODE:**

import java.util.Scanner;

class ascii

{

public static void main( String[] args)

{

System.out.println("Enter char whose ASCII code is to be found");

Scanner scanner=new Scanner(System.in);

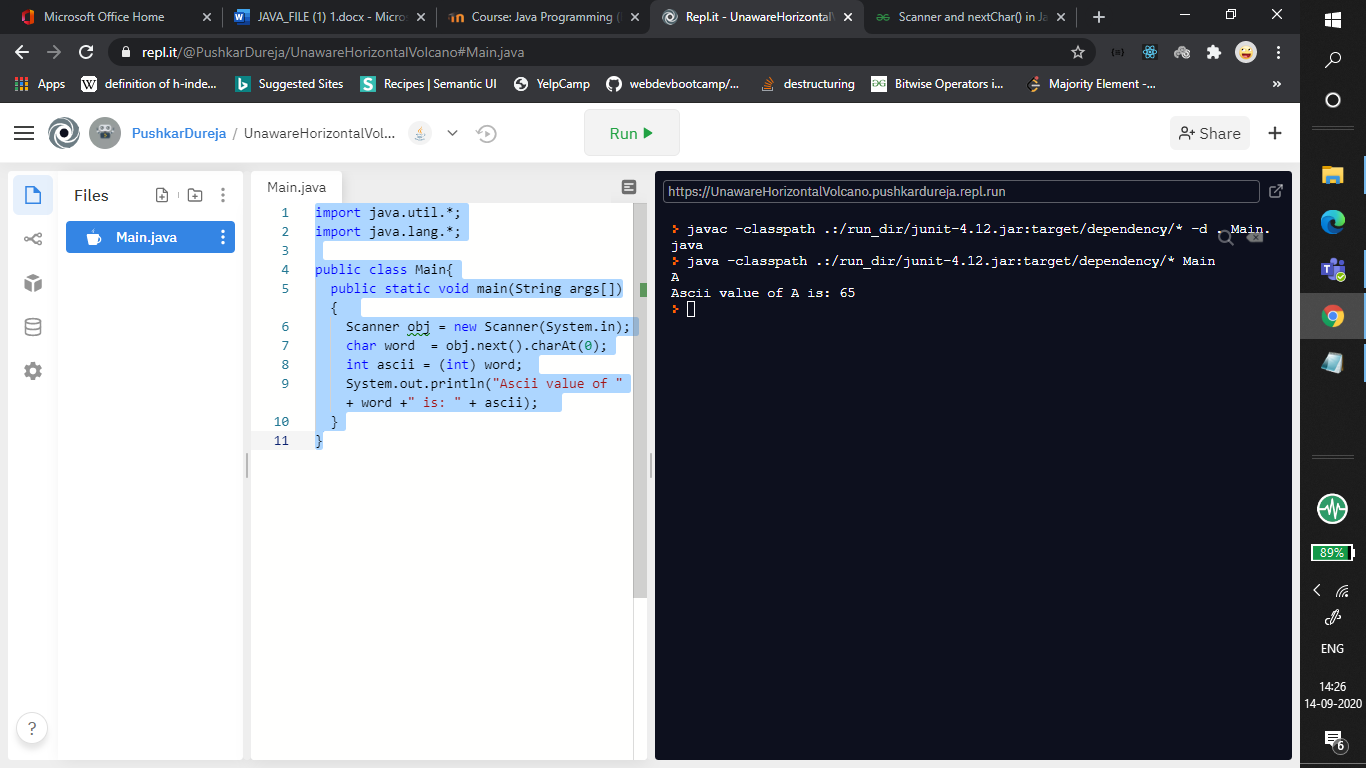
char c=scanner.next().charAt(0);

int ascii=(int) c;

System.out.println("ASCII code is "+ ascii);

}

}



**EXPERIMENT NO: 10**

**AIM:**

**Write a program in java to convert a character variable to the string type.**

**CODE:**

class chartostring

{

public static void main(String args[])

{

System.out.println("Enter a char: ");

char a;

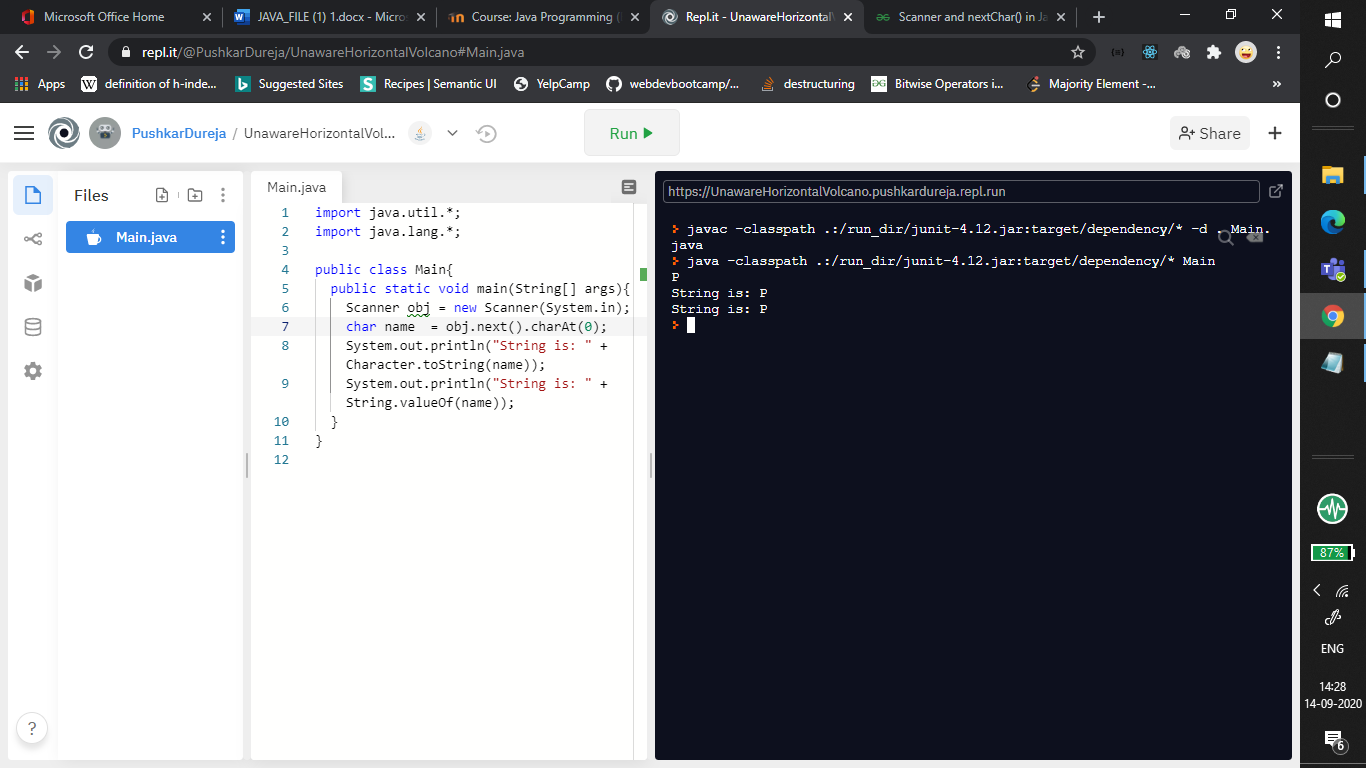
a = scn.next().charAt(0);

String s=String.valueOf(c);

System.out.println("Char is now String : "+s);

}

}



**EXPERIMENT NO: 11**

**AIM:**

**Write a java program to check if entered number is a palindrome**

**CODE:**

import java.util.Scanner;

class Palindrome{

public static void main(String args[]){

int r,sum,temp;

int n;

sum=0;

Scanner reader = new Scanner(System.in);

System.out.print("Enter the number: ");

n = reader.nextInt();

System.out.println("The number entered is:"+n);

temp=n;

while(n>0){

r=n%10; //getting remainder

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

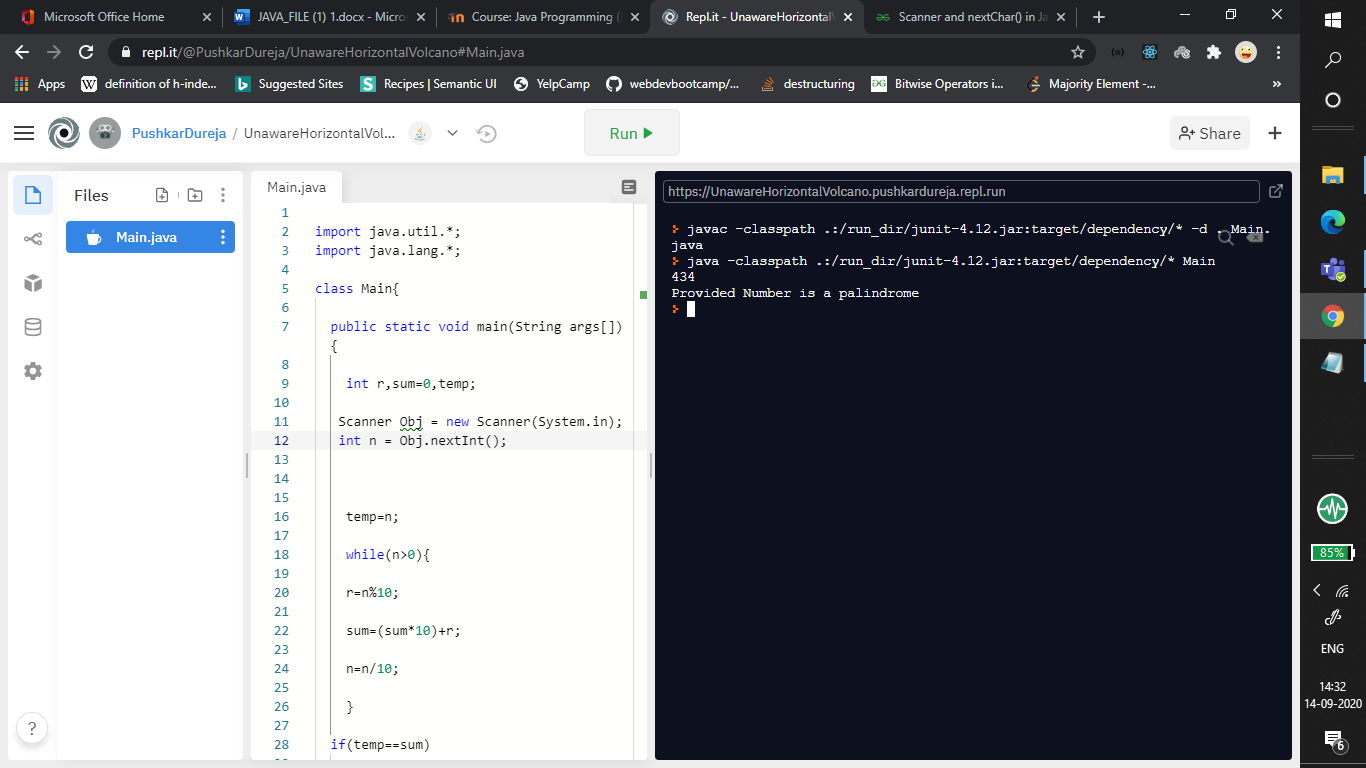
System.out.println("Number inputted is a palindrome");

else

System.out.println("not palindrome");

}

}



**EXPERIMENT NO: 12**

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

**STACK**

**SOURCE CODE**

public class StackCustom {

int size;

int arr[];

int top;

StackCustom(int size) {

this.size = size;

this.arr = new int[size];

this.top = -1;

}

public void push(int pushedElement) {

if (!isFull()) {

top++;

arr[top] = pushedElement;

System.out.println("Pushed element:" + pushedElement);

} else {

System.out.println("Stack is full !");

}

}

public int pop() {

if (!isEmpty()) {

int returnedTop = top;

top--;

System.out.println("Popped element :" + arr[returnedTop]);

return arr[returnedTop];

} else {

System.out.println("Stack is empty !");

return -1;

}

}

public int peek() {

if(!this.isEmpty())

                        return arr[top];

                else

                {

                        System.out.println("Stack is Empty");

                        return -1;

                }

}

public boolean isEmpty() {

return (top == -1);

}

public boolean isFull() {

return (size - 1 == top);

}

public static void main(String[] args) {

StackCustom StackCustom = new StackCustom(10);

StackCustom.pop();

System.out.println("=================");

StackCustom.push(10);

StackCustom.push(30);

StackCustom.push(50);

StackCustom.push(40);

System.out.println("=================");

StackCustom.pop();

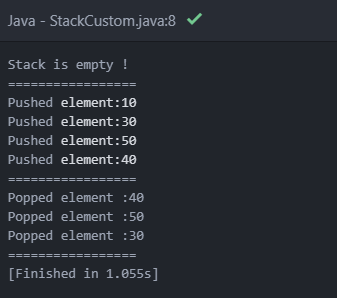
StackCustom.pop();

StackCustom.pop();

System.out.println("=================");

}

}

****

**QUEUE**

**SOURCE CODE**

import java.util.\*;

class Queue

{

private int arr[];

private int front;

private int rear;

private int capacity;

private int count;

Queue(int size)

{

arr = new int[size];

capacity = size;

front = 0;

rear = -1;

count = 0;

}

public void dequeue()

{

if (isEmpty())

{

System.out.println("UnderFlow\nProgram Terminated");

System.exit(1);

}

System.out.println("Removing " + arr[front]);

front = (front + 1) % capacity;

count--;

}

public void enqueue(int item)

{

if (isFull())

{

System.out.println("OverFlow\nProgram Terminated");

System.exit(1);

}

System.out.println("Inserting " + item);

rear = (rear + 1) % capacity;

arr[rear] = item;

count++;

}

public int peek()

{

if (isEmpty())

{

System.out.println("UnderFlow\nProgram Terminated");

System.exit(1);

}

return arr[front];

}

public int size()

{

return count;

}

public Boolean isEmpty()

{

return (size() == 0);

}

public Boolean isFull()

{

return (size() == capacity);

}

public static void main (String[] args)

{

Queue q = new Queue(5);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

System.out.println("Front element is: " + q.peek());

q.dequeue();

System.out.println("Front element is: " + q.peek());

System.out.println("Queue size is " + q.size());

q.dequeue();

q.dequeue();

if (q.isEmpty())

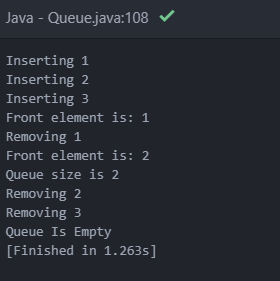
System.out.println("Queue Is Empty");

else

System.out.println("Queue Is Not Empty");

}

}

****

**EXPERIMENT NO: 13**

**AIM:**

**Write a program in java to print the factorial of a number**

**CODE:**

import java.util.Scanner;

class Factorial

{

public static void main(String args[])

{

int num,sum,temp,i;

sum=1;

Scanner reader=new Scanner(System.in);

System.out.print("Enter a number:");

num=reader.nextInt();

temp=num;

for(i=1;i<=num;i++)

{sum=sum\*temp;

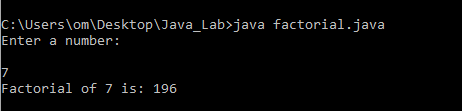
temp--;

}

System.out.println("Factorial of " + n + " is: " + f);

}

}



**EXPERIMENT NO: 14**

**AIM : Write a program to find out the array index or position where sum of numbers preceding the index the index is equals to sum of numbers succeeding the index.**

**SOURCE CODE**

class EquilibriumIndex {

int equilibrium(int arr[], int n)

{

int i, j;

int leftsum, rightsum;

for (i = 0; i < n; ++i) {

leftsum = 0;

for (j = 0; j < i; j++)

leftsum += arr[j];

rightsum = 0;

for (j = i + 1; j < n; j++)

rightsum += arr[j];

if (leftsum == rightsum)

return i;

}

return -1;

}

public static void main(String[] args)

{

EquilibriumIndex equi = new EquilibriumIndex();

int arr[] = { -7, 1, 5, 2, -4, 3, 0 };

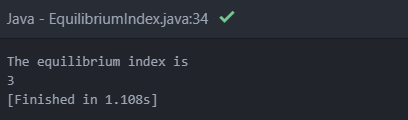
int arr\_size = arr.length;

System.out.println("The equilibrium index is ");

System.out.println(equi.equilibrium(arr, arr\_size));

}

}



**EXPERIMENT NO: 15**

**AIM : Write a program that creates and initializes a four element int array.Calculate and display the average of its values.**

**SOURCE CODE**

public class FourElementArray

{

    public static void main(String[] args) {

        int a []={11,29,333,44};

        int avg=0,sum=0;

        System.out.println("The array created is");

        for(int i=0;i<a.length;i++)

        {sum=sum+a[i];

        System.out.print(a[i]);

        System.out.print(" ");

        }

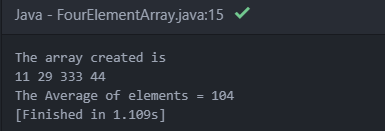
            avg=sum/(a.length);

            System.out.println();

            System.out.println("The Average of elements = "+avg);

    }

}



**EXPERIMENT NO: 16**

**AIM : WAP to sort an array.**

**SOURCE CODE**

import java.util.Arrays;

public class ArraySort

{

public static void main(String[] args)

{

// Our arr contains 8 elements

int[] arr = {13, 7, 6, 45, 21, 9, 101, 102};

    System.out.println("Unsorted Array : ");

    for(int i=0; i<arr.length; i++)

      { System.out.print(arr[i]);

        System.out.print(" ");

      }

Arrays.sort(arr);

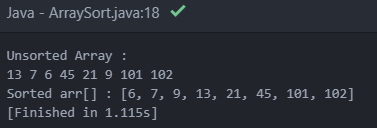
    System.out.println();

System.out.printf("Sorted arr[] : %s",

Arrays.toString(arr));

}

}

****

**EXPERIMENT NO: 17**

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

**SOURCE CODE**

class MethodOverloading

{

    void area(double x, double y)

    {

        System.out.println("the area of the triangle is "+(0.5\*x\*y)+" sq units");

    }

    void area(float x, float y)

    {

        System.out.println("the area of the rectangle is "+x\*y+" sq units");

    }

    void area(double x)

    {

        double z = 3.14 \* x \* x;

        System.out.println("the area of the circle is "+z+" sq units");

    }

}

class Overload

{

     public static void main(String[] args)

{

  MethodOverloading ob = new MethodOverloading();

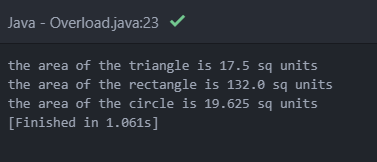
  ob.area(5.00,7.00);

  ob.area(11,12);

  ob.area(2.5);

        }

}

****

**EXPERIMENT NO: 18**

**AIM: Write a Java program to create all possible permutations of a given array of distinct integers**

**SOURCE CODE**

import java.util.\*;

import java.util.List;

public class Main {

public static void main(String[] args) throws Exception

{

int[] nums1 = {1, 2, 3, 4};

System.out.println("\nOriginal array: "+Arrays.toString(nums1));

List<List<Integer>> result1 = new Main().permute(nums1);

System.out.println("\nPossible permutations of the said array:");

result1.forEach(System.out::println);

}

public List<List<Integer>> permute(int[] nums)

{

List<List<Integer>> result = new ArrayList<>();

Permutation(0, nums, result);

return result;

}

private void Permutation(int i, int[] nums, List<List<Integer>> result) {

if (i == nums.length - 1) {

List<Integer> list = new ArrayList<>();

for (int n : nums) list.add(n);

result.add(list);

}

else

{

for (int j = i, l = nums.length; j < l; j++)

{

int temp = nums[j];

nums[j] = nums[i];

nums[i] = temp;

Permutation(i + 1, nums, result);

temp = nums[j];

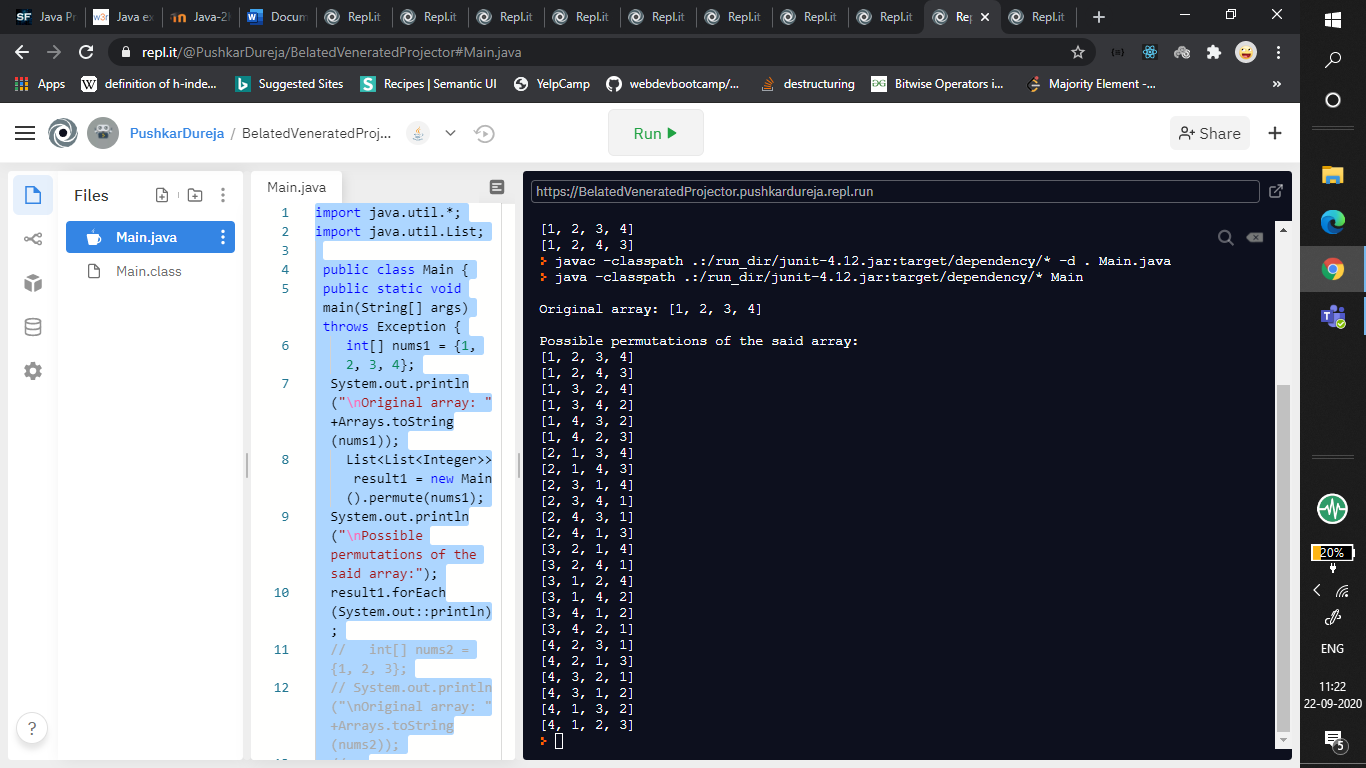
nums[j] = nums[i];

nums[i] = temp;

}

}

}

}

**EXPERIMENT NO: 19**

**AIM: Write a program that accepts a number and displays the output in word. Eg. Input – 123, output – One Hundred Twenty-Three**

**SOURCE CODE**

class Main

{

static void convert\_to\_words(char[] num)

{

int len = num.length;

if (len == 0)

{

System.out.println("empty string");

return;

}

if (len > 4)

{

System.out.println("Length more than 4 is not supported");

return;

}

String[] single\_digits = new String[]{ "zero", "one", "two", "three","four","five","six","seven",

"eight", "nine"};

String[] two\_digits = new String[]{"", "ten", "eleven", "twelve", "thirteen", "fourteen",

"fifteen", "sixteen", "seventeen","eighteen", "nineteen"};

String[] tens\_multiple = new String[]{"", "", "twenty", "thirty", "forty", "fifty","sixty", "seventy", "eighty", "ninety"};

String[] tens\_power = new String[] {"hundred", "thousand"};

System.out.print(String.valueOf(num)+": ");

if (len == 1)

{

System.out.println(single\_digits[num[0] - '0']);

return;

}

int x = 0;

while (x < num.length)

{

if (len >= 3)

{

if (num[x]-'0' != 0)

{

System.out.print(single\_digits[num[x] - '0']+" ");

System.out.print(tens\_power[len - 3]+" ");

}

--len;

}

else

{

if (num[x] - '0' == 1)

{

int sum = num[x] - '0' +

num[x] - '0';

System.out.println(two\_digits[sum]);

return;

}

else if (num[x] - '0' == 2 && num[x + 1] - '0' == 0)

{

System.out.println("twenty");

return;

}

else

{

int i = (num[x] - '0');

if(i > 0)

System.out.print(tens\_multiple[i]+" ");

else

System.out.print("");

++x;

if (num[x] - '0' != 0)

System.out.println(single\_digits[num[x] - '0']);

}

}

++x;

}

}

public static void main(String[] args)

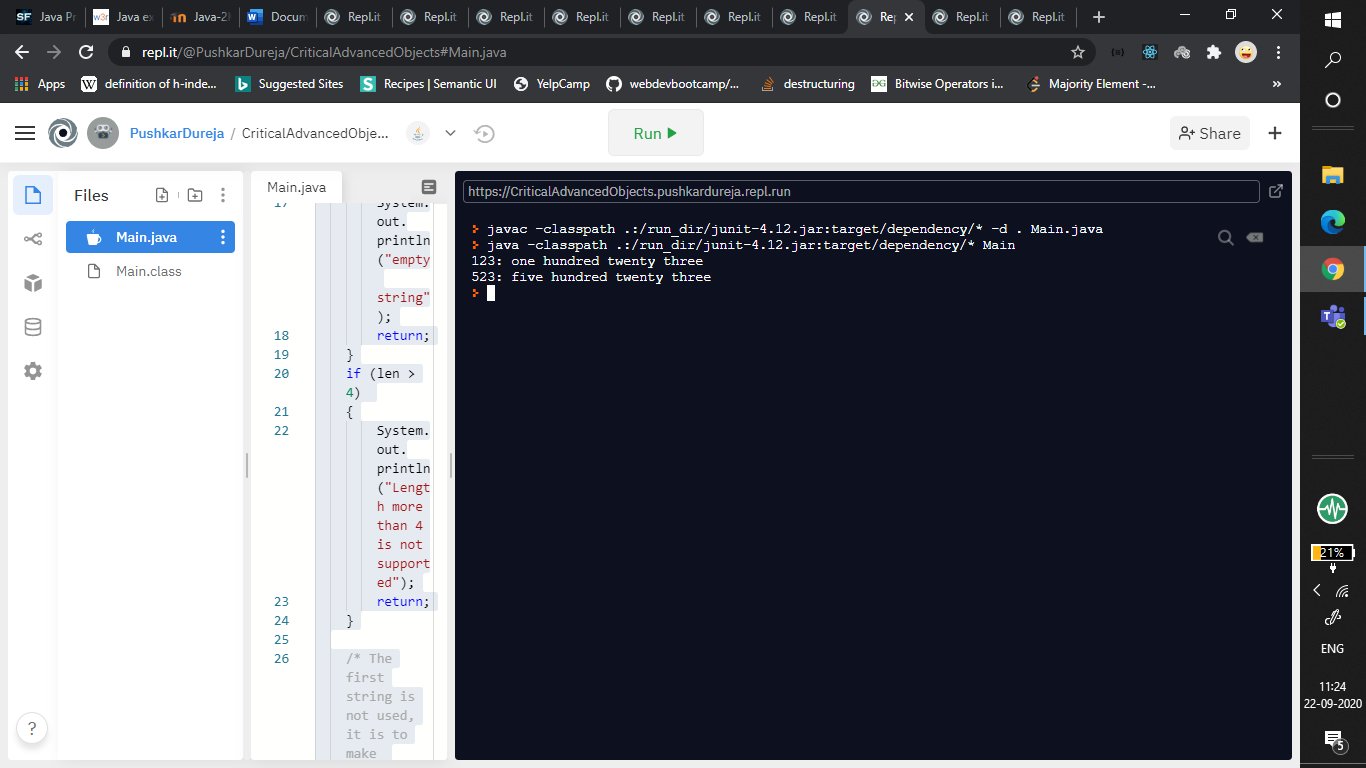
{

convert\_to\_words("123".toCharArray());

convert\_to\_words("523".toCharArray());

}

}



**EXPERIMENT NO: 20**

**AIM: Write a program to check whether a given string is Palindrome or not.**

**SOURCE CODE**

import java.util.Scanner;

public class Main

{

static boolean isPalindrome(String str)

{

int i = 0, j = str.length() - 1;

while (i < j)

{

if (str.charAt(i) != str.charAt(j))

return false;

i++;

j--;

}

return true;

}

public static void main(String[] args)

{

Scanner obj = new Scanner(System.in);

String str = obj.next();

if (isPalindrome(str))

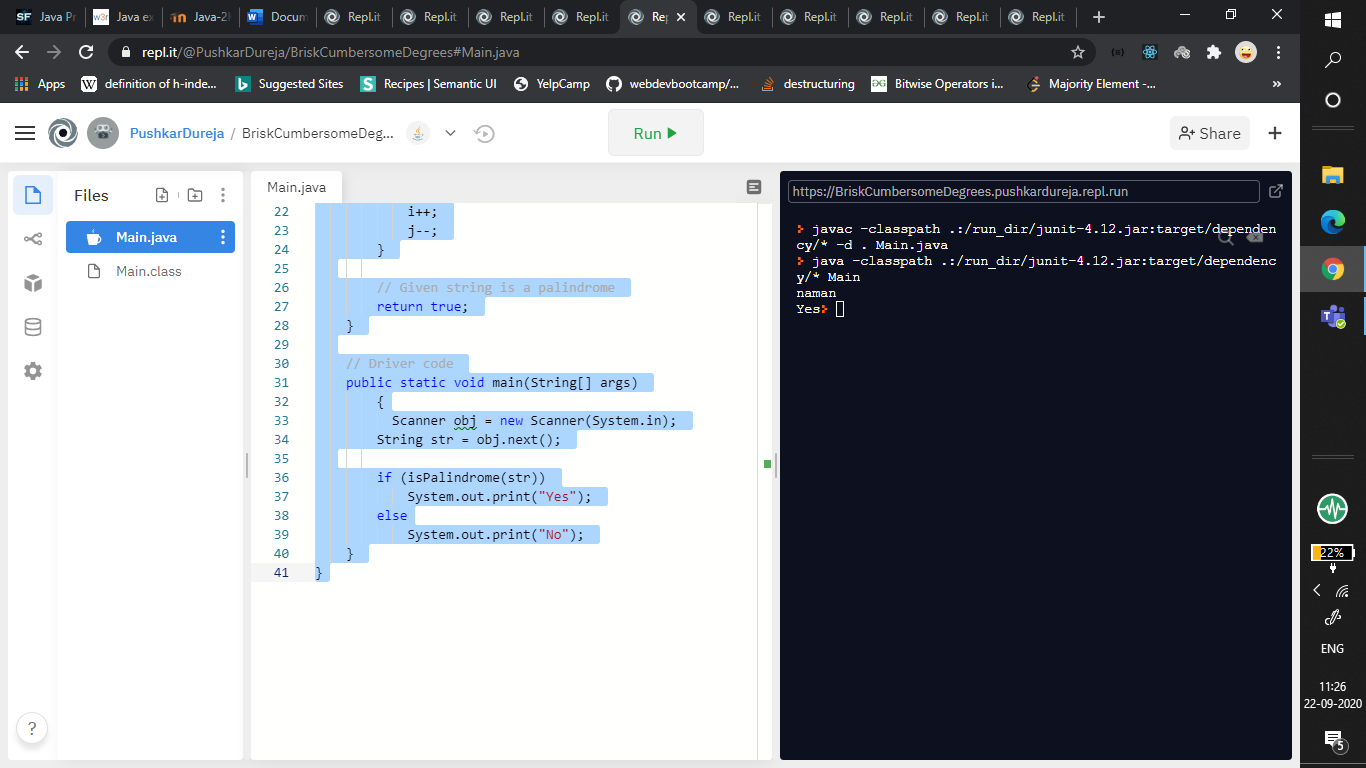
System.out.print("Yes");

else

System.out.print("No");

}

}



**EXPERIMENT NO: 21**

**AIM: Write a Java program to divide a string into n equal parts.**

**SOURCE CODE:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner obj = new Scanner(System.in);

String str = obj.next();

int len = str.length();

int n = obj.nextInt();

int temp = 0, chars = len/n;

String[] equalStr = new String [n];

if(len % n != 0) {

System.out.println("Sorry this string cannot be divided into "+ n +" equal parts.");

}

else {

for(int i = 0; i < len; i = i+chars) {

String part = str.substring(i, i+chars);

equalStr[temp] = part;

temp++;

}

System.out.println(n + " equal parts of given string are ");

for(int i = 0; i < equalStr.length; i++) {

System.out.println(equalStr[i]);

}

}

}

}



**EXPERIMENT NO: 22**

**AIM : Write a program that count the number of instances created for the class.**

**SOURCE CODE :**

public class No\_objects

{

static int count = 0;

No\_objects()

{

count++;

}

public static void main(String args[])

{

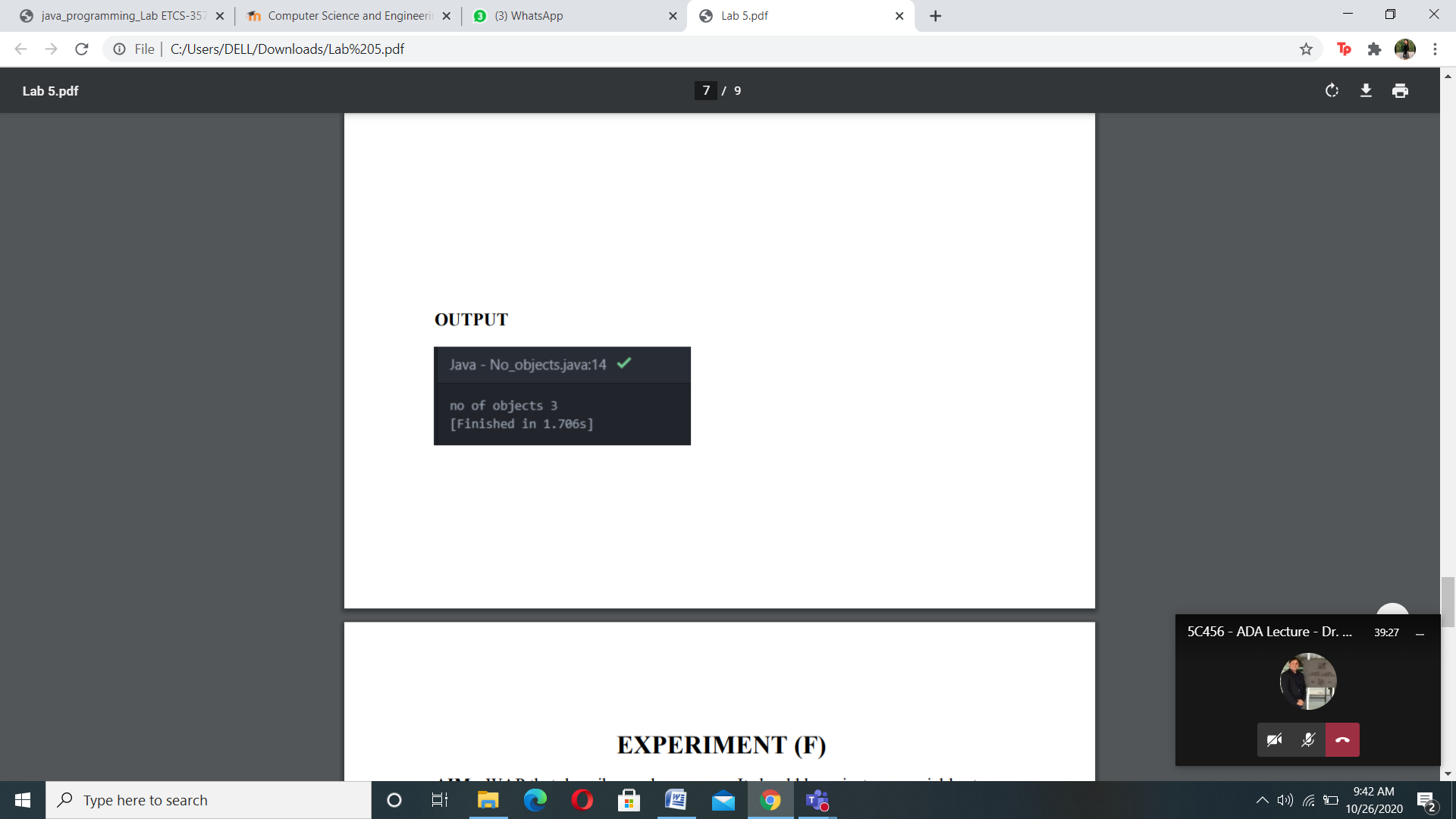
No\_objects o1 = new No\_objects();

No\_objects o2 = new No\_objects();

No\_objects o3 = new No\_objects();

System.out.println("no of objects " +count);

} }

****

**EXPERIMENT NO: 23**

**AIM : WAP that creates a class circle with instance variables for the centre and the radius . Initialize and display its variables.**

**SOURCE CODE**

public class Circle

{

public double x=5,y=5;

public double radius=1;

public static void main(String args[])

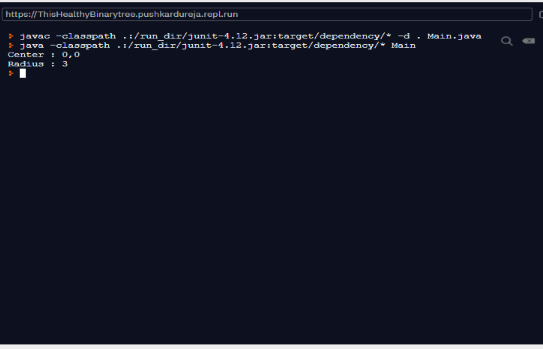
{

Circle c=new Circle();

System.out.println("center is at " +c.x +"x and "+c.y+"y");

System.out.println("radius of circle is " + c.radius);

} }



**EXPERIMENT NO: 24**

**AIM : Modify experiment 1 to show constructor overloading**

**SOURCE CODE**

public class Circle

{

private double radius;

private String color;

public Circle()

{

radius = 1.0;

color = "red";

}

public Circle(double r)

{

radius = r;

color = "red";

}

public double getRadius()

{

return radius;

}

public double getArea()

{

return radius\*radius\*Math.PI;

}

public static void main(String[] args)

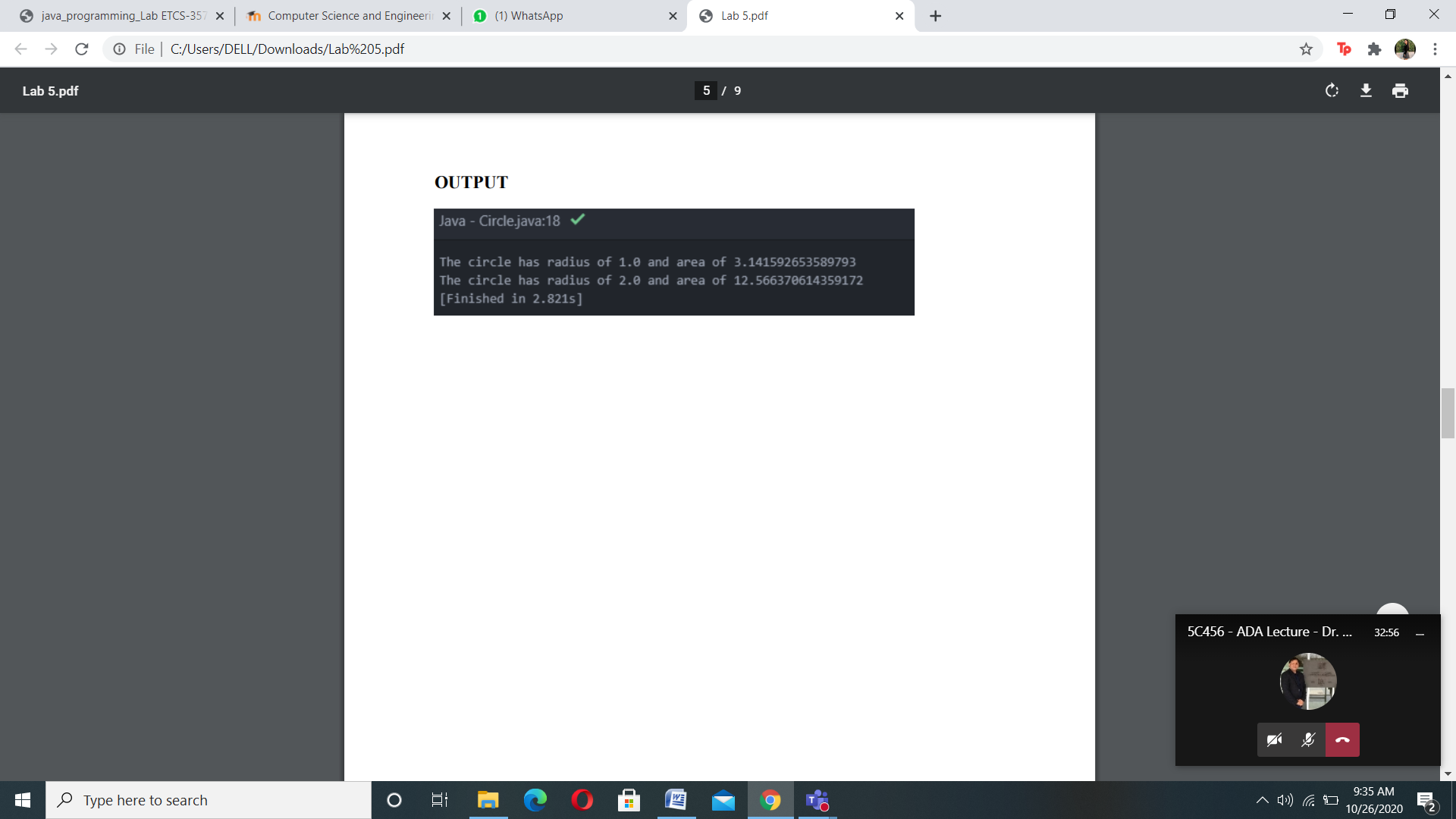
{

Circle c1 = new Circle();

System.out.println("The circle has radius of " + c1.getRadius() + " and area of " + c1.getArea()); Circle c2 = new Circle(2.0);

System.out.println("The circle has radius of " + c2.getRadius() + " and area of " + c2.getArea());

} }



**EXPERIMENT NO: 25**

**AIM: WAP that creates a class circle with instance variables for the center and the radius. It should have overloaded constructors. If the user passes values then the values should be assigned to the center co-ordinates and radius, otherwise they should be assigned any value that you find fit. It should also have a function to calculate Area and Perimeter. Also, define a plot function that can display where the center is located, what is the diameter of the circle and its area and perimeter**

**SOURCE CODE:**

public class exp1 {

public static class Circle {

int x;

int y;

int rad;

Circle() {

this.x = 0;

this.y = 0;

this.rad = 1;

}

Circle(int x, int y, int rad) {

this.x = x;

this.y = y;

this.rad = rad;

}

void getCenter(){

System.out.println("the center of the cicle is ("+this.x+","+this.y+").");

}

void getDiameter(){

System.out.println("the diameter of the circle is "+2\*this.rad+".");

}

void getPerimeter(){

System.out.println("the Permimeter of the circle is "+(2\*3.14\*this.rad)+".");

}

void getArea(){

System.out.println("the area of the circle is "+(3.14\*this.rad\*this.rad)+".");

}

}

public static void main(String[] args) {

Circle C=new Circle(10,10,20);

C.getCenter();

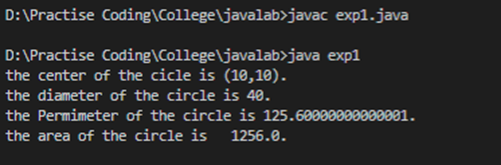
C.getDiameter();

C.getPerimeter();

C.getArea();

}

}



**EXPERIMENT NO: 26**

**AIM: Write a program that defines shape class as an abstract class. It should have Area function. Also, implement this class for any two shapes of your choice**

**SOURCE CODE:**

public class exp2 {

public static abstract class Shape {

private String id;

public Shape(String id) {

this.id = id;

}

public abstract double getArea();

public String getId() {

return id;

}

public String toString() {

return "Shape[id=" + id + ",area=" + getArea() + "]";

}

}

public static class Circle extends Shape {

private double radius;

public Circle(String name, double r) {

super(name);

radius = r;

}

// Overide the abstract method declared in shape

public double getArea() {

return Math.PI \* radius \* radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double newRadius) {

radius = newRadius;

}

public String toString() {

return "Circle => radius" + radius + "," + super.toString();

}

}

public static class Rectangle extends Shape {

private double width, height;

public Rectangle(String name, double w, double h) {

super(name);

width = w;

height = h;

}

public double getArea() {

return width \* height;

}

public double getWidth() {

return width;

}

public double getHeight() {

return height;

}

public void setWidthHeight(double newWidth, double newHeight) {

width = newWidth;

height = newHeight;

}

public String toString() {

return "Rectangle => width=" + width + ",height=" + height + "," + super.toString();

}

}

public static void main(String[] args) {

Shape C = new Circle("Circle", 20);

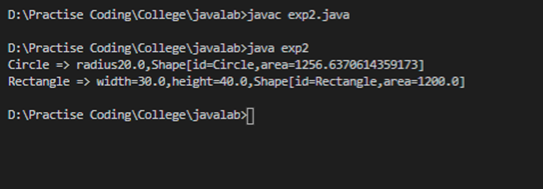
System.out.println(C.toString());

Shape R=new Rectangle("Rectangle",30,40);

System.out.println(R.toString());

}

}



**EXPERIMENT NO: 27**

**AIM: Given Class Dog.java, derive two subclasses as specific breeds of dogs. Then calculate the average weight of dog. Each class derived from dog.java should have an instance variable of weight.**

**SOURCE CODE:**

public class exp3 {

public static class Dog {

protected String name;

public Dog() {

this.name = "dog";

}

public Dog(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public String Speek() {

return "woof";

}

}

public static class Lebra extends Dog {

int wt;

Lebra(String name, int wt) {

super(name);

this.wt = wt;

}

void intro() {

System.out.println("hello i am " + super.name+ ". My weight is "+this.wt);

}

int getWt() {

return this.wt;

}

}

public static class Saint\_Bernard extends Dog {

int wt;

Saint\_Bernard(String name, int wt) {

super(name);

this.wt = wt;

}

void intro() {

System.out.println("hello i am " + super.name + ". My weight is " + this.wt);

}

int getWt() {

return this.wt;

}

}

public static int averageWt(int wt1, int wt2) {

return (wt1 + wt2) / 2;

}

public static void main(String[] args) {

Lebra L = new Lebra("Doggo", 20);

Saint\_Bernard S = new Saint\_Bernard("Cheems", 28);

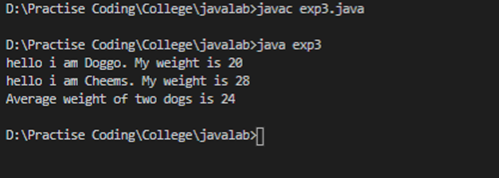
L.intro();

S.intro();

System.out.println("Average weight of two dogs is "+averageWt(L.wt, S.wt));

}

}



**EXPERIMENT NO: 28**

**AIM : Create a class person which contains basic information like name, age, phone number etc. Now derive a class employee which contains employee information like employee number, employee name, phone number etc. Now write a driver class which can compare two employees to show whether the employee is same or not. Hint: Use the equals method, which is inherited from the Object class, to determine whether two players are the same. You will need to override this method so that it gives the right result.**

**SOURCE CODE**

public class exp5 {

public static class Person {

String name;

int age;

int number;

Person() {

this.name = "";

this.age = 0;

}

Person(String name, int age, int number) {

this.name = name;

this.age = age;

this.number = number;

}

}

public static class Employee extends Person {

Employee(String name, int age, int number) {

super(name, age, number);

}

@Override

public boolean equals(Object o) {

if (o == this) {

return true;

}

if (!(o instanceof Employee)) {

return false;

}

Employee c = (Employee) o;

return name.equals(c.name) && Integer.compare(age, c.age) == 0 && Integer.compare(number, c.number) == 0;

}

}

public static void main(String[] args) {

Employee e1 = new Employee("Tom", 22, 9900);

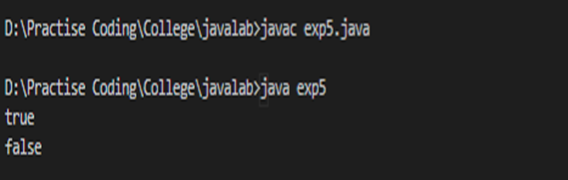
Employee e2 = new Employee("Jerry", 16, 8790);

System.out.println(e1.equals(e1));

System.out.println(e1.equals(e2));

}

}



**EXPERIMENT NO: 29**

**AIM :** **Implement the following classes. Author class will have no member functions. Only private variables.**

**Assume that a book is written by one (and exactly one) author. The Book class (as shown in the class diagram) contains the following members:**

* **Four private member variables: name (String), author (an instance of the Author class we have just created, assuming that each book has exactly one author), price (double), and qty (int).**
* **The public getters and setters: getName(), getAuthor(), getPrice(), setPrice(), getQty(), setQty().**
* **A toString() that returns "'book-name' by author-name (gender) at email". You could reuse the Author's toString() method, which returns "author-name (gender) at email". Write the driver program to test the functions.**

**SOURCE CODE:**

import java.io.\*;

class Author {

private String name;

private String email;

private char gender;

public Author(String name, String email, char gender) {

this.name = name;

this.email = email;

this.gender = gender;

}

public String getName() {

return name;

}

public char getGender() {

return gender;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String toString() {

return name + " (" + gender + ") at " + email;

}

}

class Book {

private String name;

private Author author;

private double price;

private int qty;

public Book(String name, Author author, double price, int qty) {

this.name = name;

this.author = author;

this.price = price;

this.qty = qty;

}

public String getName() {

return name;

}

public Author getAuthor() {

return author;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getQty() {

return qty;

}

public void setQty(int qty) {

this.qty = qty;

}

public String toString() {

return "'" + name + "' by " + author;

}

}

class Main {

public static void main(String[] args) {

Author ahTeck = new Author("Tan Ah Teck", "[ahTeck@somewhere.com](mailto:ahTeck@somewhere.com)", 'm');

System.out.println(ahTeck);

Book dummyBook = new Book("Java for dummies", ahTeck, 9.99, 99);

System.out.println(dummyBook);

dummyBook.setPrice(8.88);

dummyBook.setQty(88);

System.out.println("name is: " + dummyBook.getName());

System.out.println("price is: " + dummyBook.getPrice());

System.out.println("qty is: " + dummyBook.getQty());

System.out.println("author is: " + dummyBook.getAuthor());

System.out.println("author's name is: " + dummyBook.getAuthor().getName());

System.out.println("author's email is: " + dummyBook.getAuthor().getEmail());

System.out.println("author's gender is: " + dummyBook.getAuthor().getGender());

Book moreDummyBook = new Book("Java for more dummies",

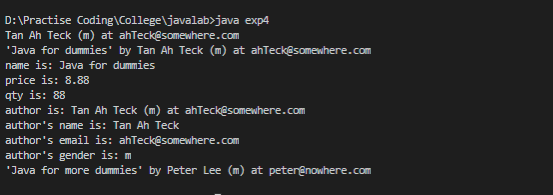
new Author("Peter Lee", "[peter@nowhere.com](mailto:peter@nowhere.com)", 'm'),

19.99, 8);

System.out.println(moreDummyBook);

}

}



**EXPERIMENT NO: 30**

**AIM :** **Implement the following classes**

* **Circle**
* **Cylinder**

**Write the driver program to test the cylinder class.**

**SOURCE CODE:**

class Circle

{

private double radius;

private String color;

public Circle()

{

radius = 1.0;

color = "red";

}

public Circle(double r)

{

radius = r;

color = "red";

}

public double getRadius()

{

return radius;

}

public void setRadius(double newRadius) {

radius = newRadius;

}

public double getArea() {

return radius\*radius\*Math.PI;

}

public void setColor(String newColor) {

color= newColor;

}

}

class Cylinder extends Circle {

private double height;

public Cylinder()

{

super();

height = 1.0;

}

public Cylinder(double height) {

super();

this.height = height;

}

public Cylinder(double radius, double height) {

super(radius);

this.height = height;

}

public double getHeight() {

return height;

}

public double getVolume() {

return getArea()\*height;

}

public String toString()

{

return "Cylinder: subclass of " + super.toString() + " height=" + height;

}

}

class Main {

public static void main (String[] args) {

Cylinder c1 = new Cylinder();

System.out.println("Cylinder:"+ " radius=" + c1.getRadius()+ " height=" + c1.getHeight()

+ " base area=" + c1.getArea()+ " volume=" + c1.getVolume());

Cylinder c2 = new Cylinder(10.0);

System.out.println("Cylinder:"+ " radius=" + c2.getRadius()+ " height=" + c2.getHeight()

+ " base area=" + c2.getArea()+ " volume=" + c2.getVolume());

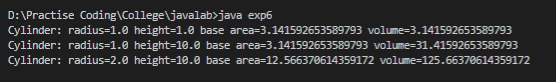
Cylinder c3 = new Cylinder(2.0, 10.0);

System.out.println("Cylinder:"+ " radius=" + c3.getRadius()+ " height=" + c3.getHeight()

+ " base area=" + c3.getArea()+ " volume=" + c3.getVolume());

}

}



**EXPERIMENT NO: 31**

**AIM : WAP to implement simple Inheritance**

**SOURCE CODE:**

public class inheritance **{**

public static class Person {

String name;

int age;

Person() {

this.name = "hello";

this.age = 20;

}

void getName() {

System.out.println(this.name);

}

}

public static class Person2 extends Person {

Person2() {

super();

}

void getAge(){

System.out.println(this.age);

}

}

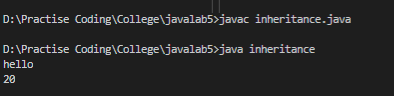
public static void main(String[] args){

Person2 p=new Person2();

p.getName();

p.getAge();

}  
}



**EXPERIMENT NO: 32**

**Aim: WAP to implement method overriding**

**SOURCE CODE:**

public class methodoveriding{

/// METHOD OVERIDING

public static class Person{

String name;

int age;

Person(){

this.name="hello";

this.age=20;

}

void getName(){

System.out.println(this.name);

}

}

public static class Person2 extends Person{

Person2(){

super();

}

void getName(){

System.out.println("name => "+this.name +" age => "+this.age);

}

}

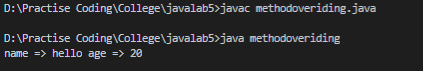
public static void main(String[] args){

Person2 p=new Person2();

p.getName();

}

**}**



**EXPERIMENT NO: 33**

**Aim: WAP to implement multiple Inheritance**

**CODE:**

public class multilevel {

public static class Person {

String name;

int age;

Person() {

this.name = "hello";

this.age = 20;

}

void getName() {

System.out.println(this.name);

}

}

public static class Person2 extends Person {

Person2() {

super();

}

void getName() {

System.out.println("name => " + this.name + " age => " + this.age);

}

void getOnlyAge(){

System.out.println(this.age);

}

}

public static class Person3 extends Person2 {

Person3() {

super();

}

void getDescription(){

System.out.println("I am Person3 extending Person1 and Person2");

}

}

public static void main(String[] args ){

Person3 p=new Person3();

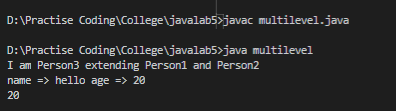
p.getDescription();

p.getName();

p.getOnlyAge();

}

}



**EXPERIMENT NO: 34**

**AIM: WAP to implement different functions of super keyword**

**SOURCE CODE:**

public class AllSuper {

public static class Person {

String name;

int age;

Person() {

this.name = "hello";

this.age = 20;

}

Person(String name, int age) {

this.name = name;

this.age = age;

}

void getName() {

System.out.println(this.name);

}

}

public static class Person2 extends Person {

Person2() {

super();

}

Person2(String name, int age) {

super(name, age);

}

void getName() {

super.getName();

}

}

public static void main(String[] args) {

Person p1=new Person2();

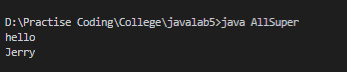
p1.getName();

Person2 p2=new Person2("Jerry",20);

p2.getName();

}

}



**EXPERIMENT NO: 35**

**AIM: Write a program in java programming language that illustrates a interface and implements it**

**SOURCE CODE:**

interface MyInterface

{

public void method1();

public void method2();

}

public class first implements MyInterface

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[])

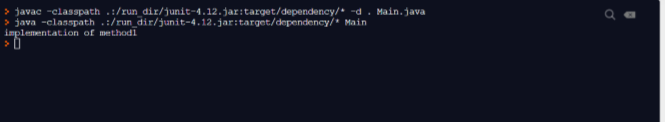
{

MyInterface obj = new first();

obj.method1();

}

}



**EXPERIMENT NO: 36**

**AIM: Write a program in java programming language that illustrates a interface and implements it using single inheritance**

**SOURCE CODE:**

interface Inf1{

public void method1();

}

interface Inf2 extends Inf1 {

public void method2();

}

public class first implements Inf2{

public void method1(){

System.out.println("method1");

}

public void method2(){

System.out.println("method2");

}

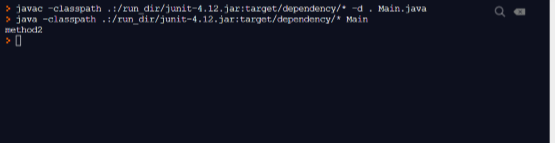
public static void main(String args[]){

Inf2 obj = new first();

obj.method2();

}

}



**EXPERIMENT NO: 37**

**AIM:Write a program in java programming language that illustrates a interface and implements it using multilevel inheritance**

**SOURCE CODE:**

interface vehicleone{

int speed=90;

public void distance();

}

interface vehicletwo{

int distance=100;

public void speed();

}

class Vehicle implements vehicleone,vehicletwo{

public void distance(){

int distance=speed\*100;

System.out.println("distance travelled is "+distance);

}

public void speed(){

int speed=distance/100;

System.out.println("speed is "+speed);

}

}

public class MultipleInheritanceUsingInterface{

public static void main(String args[]){

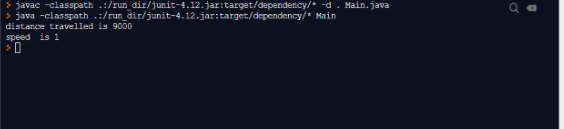
Vehicle obj= new Vehicle();

obj.distance();

obj.speed();

}

}



**EXPERIMENT NO:38**

**AIM: Write an application that executes two threads. One thread displays “A” every 1000 milliseconds and other displays “B” every 3000 milliseconds. Create the threads by extending the Thread class.**

**SOURCE CODE:**

class ThreadExample extends Thread

{

ThreadExample(String s)

{

super(s);

start();

}

public void run()

{

for(int i=0;i<10;i++)

{

System.out.println(Thread.currentThread().getName());

try

{

if(Thread.currentThread().getName()=="A")

{

Thread.sleep(1000);

}

else

{

Thread.sleep(3000);

}

}

catch(Exception e){}

}

}

}

class q1

{

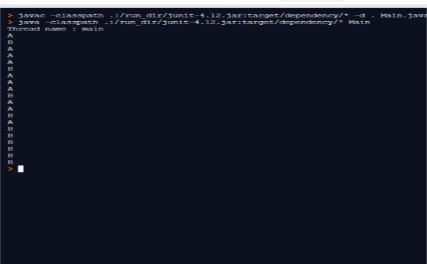
public static void main(String arg[])

{

System.out.println("Thread name : "+Thread.currentThread().getName());

ThreadExample e1=new ThreadExample("A");

ThreadExample e2=new ThreadExample("B"); } }



**EXPERIMENT NO: 39**

**AIM: Write a Java program that implements a multi-threaded program has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd the third thread will print the value of cube of the number.**

**SOURCE CODE:**

import java.util.Random;

class RandomNumberThread extends Thread {

public void run() {

Random random = new Random();

for (int i = 0; i < 10; i++) {

int randomInteger = random.nextInt(100);

System.out.println("Random Integer generated : " + randomInteger);

if((randomInteger%2) == 0) {

SquareThread sThread = new SquareThread(randomInteger);

sThread.start();

}

else {

CubeThread cThread = new CubeThread(randomInteger);

cThread.start();

}

try {

Thread.sleep(1000);

}

catch (InterruptedException ex) {

System.out.println(ex);

}

}

}

}

class SquareThread extends Thread {

int number;

SquareThread(int randomNumbern) {

number = randomNumbern;

}

public void run() {

System.out.println("Square of " + number + " = " + (number \* number));

}

}

class CubeThread extends Thread {

int number;

CubeThread(int randomNumber) {

number = randomNumber;

}

public void run() {

System.out.println("Cube of " + number + " = " + number \* number \* number);

}

}

public class q5 {

public static void main(String args[]) {

RandomNumberThread rnThread = new RandomNumberThread();

rnThread.start();

}

}

