## CHAROTAR UNIVERSITY OF SCIENCE TECHNOLOGY

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY RESEARCH**

**Department of Information Technology**

**Subject : JAVA PROGRAMMING**

**Subject Code: IT267 Semester: III**

**Academic Year :2023-24(ODD)**

**PART-I**

## Data Types, Variables,String,Control Statements,Operators,Array

**[1.] Aim :-**

Introduction to Object Oriented Concepts, comparison of Java with other object- oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument. Introduction to Eclipse or Netbeans IDE, or BlueJ and Console Programming**.**

## CODE :-

**Object Oriented Concepts:**

 Class

 Object

 Encapsulation

 Inheritance

 Polymorphism

 Abstraction

##  Comparison of Java with other object-oriented programminglanguages.

|  |  |
| --- | --- |
| **Java** | **C++** |
| No use of pointers. Supports references,  thread and interfaces. | Use of pointers, structures, union. |
| Wide range of classes for various high  level services. | Comparatively available with low level  functionalities. |
| Multiple inheritance is partially done  through interfaces. | Provide both single and multiple  inheritance. |
| It doesn’t support this feature. | Supports operator overloading. |
| Platform independent, byte code  generated works on every OS. | Platform dependent, must be  recompiled for different platform. |

|  |  |
| --- | --- |
| **Java** | **Python** |
| Java is both compiled and interpreted  language, which is first compiled and then interpreted into a byte code. | Python is an interpreted programming language. |
| Define particular block by curly braces,  end statements by ; | No need of semi colons and curly  braces, uses indentation. |
| Strongly typed, need to define the exact  datatype of variables. | Dynamic, no need to define the exact  datatype of variables. |
| Multiple inheritance is partially done  through interfaces. | Provide both single and multiple  inheritance. |
| Java is much faster than python in terms  of speed. | Expected to run slower than Java  programs. |

 **Introduction to JDK, JRE, JVM, Javadoc :-**

 **JDK** – **Java Development Kit** (in short JDK) is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) which includes two things

1. Development Tools (to provide an environment to develop your java programs)
2. JRE (to execute your java program).

 **JRE** – **Java Runtime Environment** (to say JRE) is an installation packagewhich provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

 **JVM** - **Java Virtual machine** (JVM) is a very important part of both JDKand JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible

for **executing the java program line by line** hence it is also known as interpreter.

**javadoc - javadoc** tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set of source file describing classes, methods, constructors, and fields.

##  Command line argument.

 The java command-line argument is an argument i.e. passed at the time of running the java program.

 The arguments passed from the console can be received in the java program and it can be used as an input.

##  Introduction to Eclipse or NetBeans IDE, or BlueJand Console Programming.

 **Introduction to Eclipse:** We use application Eclipse to edit, test, and run Java programs. It is called an IDE (Integrated Development Environment. Inthis little module, we introduce you to the basics of Eclipse. You will not completely understand the Java code we write; just concentrate on how we are using Eclipse. Besides showing you how to use the IDE in a basic fashion, we give instructions on how to set various preferences, e.g. ensuringthat line numbers appear in the editing panel.

 **Introduction to NetBeans:** NetBeans IDE is a free, open source, integrateddevelopment environment (IDE) that enables you to develop desktop, mobile and web applications. The IDE supports application development in various languages, including Java, HTML5, PHP and C++. The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. The IDE runson Windows, Linux, Mac OS X, and other UNIX-based systems.

 **Introduction to BlueJ:** BlueJ is a windows based platform for Java Development Kit (JDK). It is a free Java environment started in 1999 by

Michael Kolling and John Rosenberg at Monash University, Australia, as asuccessor to Blue. It requires to install JDK version 1.3 or more before installing BlueJ. It can be freely downloaded from its official website of BlueJ.

 **Introduction to Console Programming:** A console application is

a program which runs in an command prompt window. An example of a console application is below: Console programs do not have the flash, nor the event-driven capabilities of a Windows application, however, they still have their place.

## [2.] Aim :-

Write a program that declares one integer variable called var1. Give value 10 to this variable and then, using one println() statement, display the value on the screen like this: “10 is the value of var1.”

## Program code :-

public class prac

{

public static void main(String[] args)

{

int var1=20;

System.*out*.println(var1+" is the value of

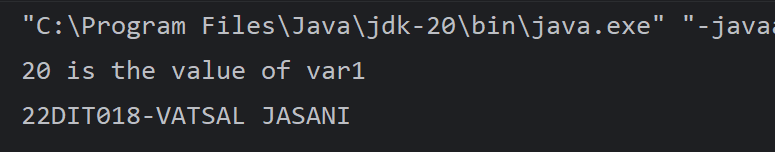
var1");

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

* After performing the practical we learn to assign value to variable and printit using System.out.println() function.

## Aim :-

Java program that converts a string entered by the user to Morse code or vice versa. It will require the implementation of data structures, including arrays, loops, and conditional statements. • Create two arrays - one to contain the strings of letters to be converted, and one to contain the Morse codes.

* In the program's main method, prompt the user for input to choose between the string or Morse.
* For Morse code conversion, read in a string from the user; use conditional statements, looping, and array methods to convert the string to Morse-code.
* For string conversion, read in a Morse-coded string from the user; use arrays, conditional statements, and looping to convert Morse code to a string

## Program code :-

import java.util.\*;

public class prac

{

public static void morsetoeng(String[] code, String morse)

{

String[] array = morse.split(" ");

System.*out*.print("Morse code "+morse+" to English is :- "); for (int i = 0; i < array.length; i++)

{

for (int j = 0; j < code.length; j++)

{

if (array[i].compareTo(code[j])==0)

{

System.*out*.print((char) (j + 'a') + " "); break;

}

}

}

}

public static void engtomorse(String[] code, String english, char[] letter)

{

System.*out*.println("\n");

System.*out*.print("Morse code of "+english+" is :- ");

for (int i = 0; i < english.length(); i++) { for (int j = 0; j < letter.length; j++) {

if (english.charAt(i)==letter[j]) {

System.*out*.print(code[j] + " "); break;

}

}

}

}

public static void main(String[] args) {

char[] letter = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',

'i', 'g', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',

'z','0','1','2','3','4','5','6','7','8','9'};

String[] code = { ".-", "-...", "-.-.", "-..", ".", "..-.",

"--.", "....", "..", ".---", "-.-", ".-..", "--",

"-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-

", "...-", ".--", "-..-", "-.--", "--..","-----",".----","..---

","...--","....-",".....","-....","--...","---..",};

Scanner sc = new Scanner(System.*in*);

System.*out*.print("enter your english word to convert into morse code : ");

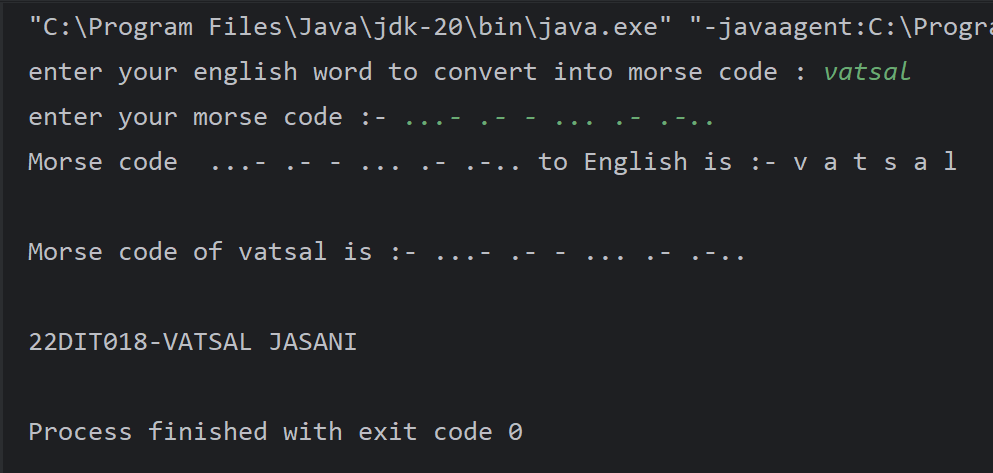
String english = sc.nextLine(); System.*out*.print("enter your morse code :-"); String morse = sc.nextLine(); *morsetoeng*(code, morse);

*engtomorse*(code, english, letter); System.*out*.println("\n"); System.*out*.println("22DIT018-VATSAL JASANI");

}

}

## Output :-



**Conclusion :-**

We got to know about the how we can split a string using java inbuilt function ‘.split’. We also got to know that how can we easily know the length of the string by the ‘.length’ function of String class. We also came across the function ‘.compareto()’ through which we could compare two strings and make our task easier

## Aim :-

A typical mobile number in India is “+91-AA-BBB-CCCCC”. Where the first two digits (AA)indicate a mobile system operator, the next three (BBB) denote the mobile switching code(MSC) while the remaining five digits (CCCCC) are unique to the subscriber. Write an application that takes a mobile number as an input from a user in above mentioned format and display code for mobile system operator, mobile switching code and last 5 digits which are unique to subscriber. Ex. For an input +91-94-999-65789, output should be :Mobile system operator code is 94 MSC is 999 Unique code is 65789 .

## Program code:

import java.util.Scanner; class prac

{

public static void main(String []args)

{

Scanner sc=new Scanner(System.*in*); System.*out*.print("enter a your mobile number: +91-"); String a=sc.nextLine();

String[] i=a.split("-",3);

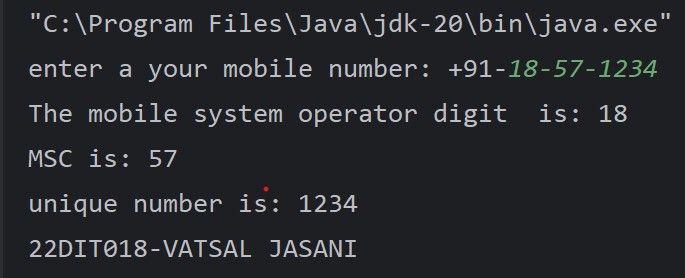
System.*out*.println("The mobile system operator digit is: "+i[0]);

System.*out*.println("MSC is: " +i[1]); System.*out*.println("unique number is: " +i[2]); System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output:**



## Conclusion :-

We got to know about the how we can split a string using java inbuilt function ‘.split’. Also we got to know about various parameters of split function and how we can apply it to arrays.

## [5.] Aim :-

An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.

## Program code :-

import java.util.Scanner;

public class prac {

public static void main(String[] args) { Scanner sc = new Scanner(System.*in*);

System.*out*.println("Motor\t\t:\t1"); System.*out*.println("Fan\t\t\t:\t2"); System.*out*.println("Tubelight\t:\t3"); System.*out*.println("Wires\t\t:\t4"); System.*out*.println("Other\t\t:\t5"); System.*out*.println("Exit\t\t:\t6"); int a = 0;

double taxm = 0, taxf = 0, taxt = 0, taxw = 0, taxo = 0; int totalm = 0, totalf = 0, totalt = 0, totalw = 0, totalo =

0;

int m = 0, f = 0, t = 0, w = 0, o = 0;

int[] price = {500, 400, 300, 200, 100};

while (a < 6) {

System.*out*.print("Enter code\t:\t"); a = sc.nextInt();

switch (a) {

case 1 -> {

totalm += price[a - 1]; taxm += 300\*0.08;

m++;

}

case 2 -> {

totalf += price[a - 1]; taxf += 500\*0.12;

f++;

}

case 3 -> {

totalt += price[a - 1]; taxt += 100\*0.04;

t++;

}

case 4 -> {

totalw += price[a - 1]; taxw += 50\*0.075;

w++;

}

case 5 -> {

totalo += price[a - 1]; taxo += 100\*0.03;

o++;

}

}

}

System.*out*.println("\nYour bill is\t:\t\n"); System.*out*.println("Total pieces of is Motor:\t" + m); System.*out*.println("Bill of motor is\t:\t" + (totalm+taxm)); System.*out*.println("\nTotal pieces of Fan is:\t" + f); System.*out*.println("Bill of fan is\t:\t" + (totalf+taxf)); System.*out*.println("\nTotal pieces of Tubelight is:\t" + t); System.*out*.println("Bill of tubelight is\t:\t" +

(totalt+taxt));

System.*out*.println("\nTotal pieces of wires is:\t" + w); System.*out*.println("Bill of wires is\t:\t" + (totalw+taxw)); System.*out*.println("\nTotal other things are:\t" + o); System.*out*.println("Bill of others is\t:\t" +

(totalo+taxo));

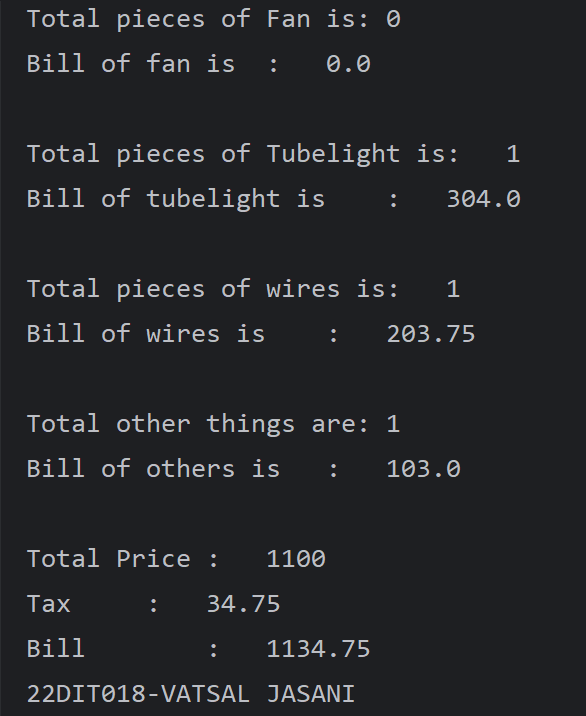
double tax = taxm + taxo + taxf + taxt + taxw;

int total = totalm + totalf + totalo + totalt + totalw; System.*out*.println("\nTotal Price\t:\t" + total); System.*out*.println("Tax\t\t:\t" + tax); System.*out*.println("Bill\t\t:\t" + (total+tax)); System.*out*.println("22DIT018-VATSAL JASANI\n\n");

}

}

**Output** :-



## Conclusion :-

The Java *switch statement* executes one statement from multiple conditions. It is like [if-else-if](https://www.javatpoint.com/java-if-else) ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use [strings](https://www.javatpoint.com/java-string) in the switch statement.

## [6.] Aim :-

Create a Java program that simulates a guessing game, where the computer picks a random number between 1 and 100 and the user has to guess it. We can use the Scanner class to 1 get user input and a loop to allow multiple guesses.

* Prompt the user to guess the number and keep track of the number of attempts they make.
* Use if-else statements to give feedback like too low or too high compared to the number.
* Use a loop to allow the user to guess again until they guess the correct number

## Program code :-

import java.util.Random; import java.util.Scanner;

public class prac {

public static void main(String[] args) { Random comp = new Random();

int n = 100;

int i = comp.nextInt() % n; int random = 1 + i; *Simulator* (random);

}

public static void Simulator (int comp) { Scanner sc = new Scanner(System.*in*);

System.*out*.print("Guess the correct number\t:\t"); while (true) {

int ui = sc.nextInt(); if (ui == comp) {

System.*out*.println("Hooray! you have guessed correct

number...");

break;

} else {

System.*out*.println("Your guess is wrong try

again...");

if (comp > ui) {

System.*out*.println("The number is greater than

input...");

} else {

System.*out*.println("The number is less than

input...");

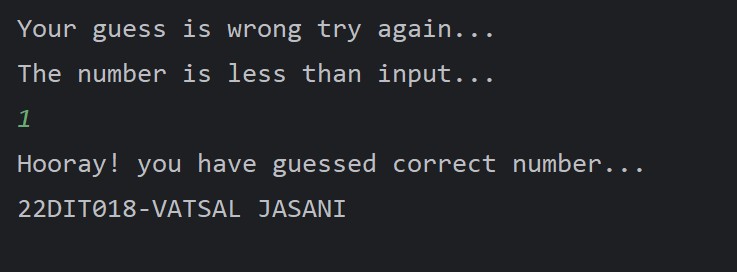
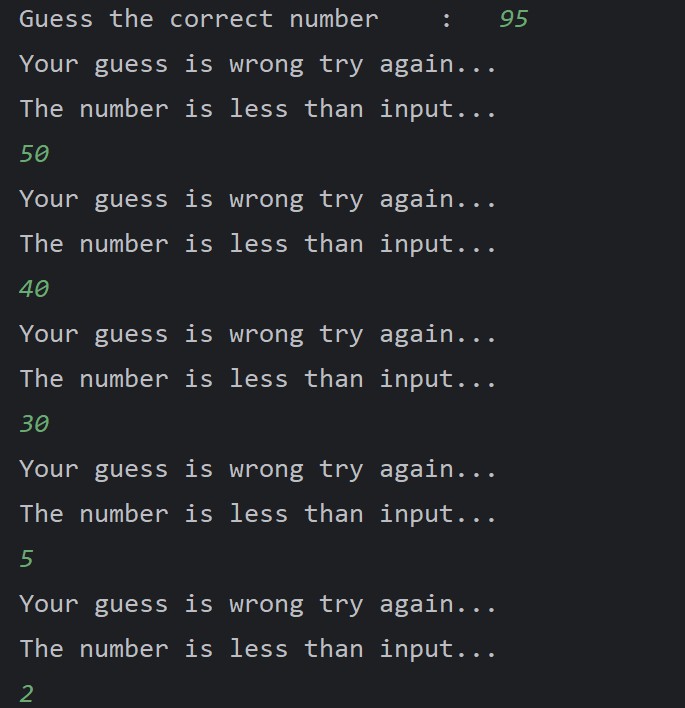
}

}

}

System.*out*.println("22DIT018-VATSAL JASANI");} }

**Output :-**



## Conclusion :-

We built a game using the Computer. We learnt about the Random class and how to use it and what are the different applications where we can use it.

**PART-II**

## [7.] Aim :-

Given a string and a non -negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front\_times('Chocolate', 2) → 'ChoCho' front\_times('Chocolate', 3) → 'ChoChoCho'

front\_times('Abc', 3) → 'AbcAbcAbc'

## Program code :-

import java.lang.String; import java.util.Scanner; class prac

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.*in*); System.*out*.print("Enter your string:"); String ch=sc.nextLine(); System.*out*.println("Enter your number:"); int h=sc.nextInt();

*front\_times*(ch, h);

System.*out*.println("\n 22DIT018-VATSAL JASANI");

}

public static void front\_times(String st,int n)

{

int l=st.length(); String s;

if(l<4)

{

s=st;

}

else{

s=st.substring(0, 3);

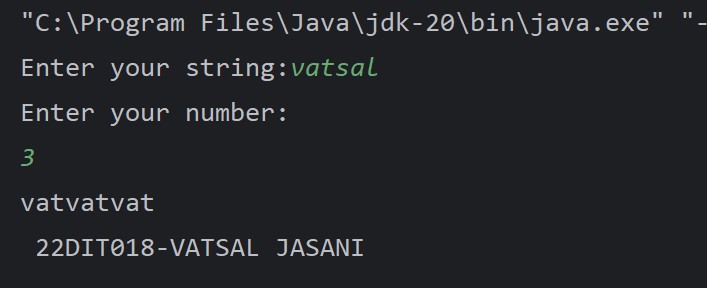
}

for(int i=0;i<n;i++) System.*out*.print(s);

}

}

**Output :-**



## Conclusion :-

Substring in [Java](https://www.simplilearn.com/tutorials/java-tutorial/what-is-java) is a commonly used method of java.lang.String class that is used to create smaller strings from the bigger one. As [strings](https://www.simplilearn.com/tutorials/java-tutorial/java-strings) are immutable in Java, the original string remains as it is, and the method returns a new string.

## [8.] Aim :-

Given an array of ints, return the number of 9's in the array. array\_count9([1, 2, 9]) → 1

array\_count9([1, 9, 9])→ 2 array\_count9([1, 9, 9, 3, 9]) → 3

## Program code :-

import java.lang.String; import java.util.Scanner; public class prac

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.*in*); System.*out*.print("Enter array size:"); int b=sc.nextInt();

int[] a=new int[b];

System.*out*.println("Enter the elements of array"); for(int i=0;i<b;i++)

{

a[i]=sc.nextInt();

}

*array\_count9*(a,b); System.*out*.println("22DIT018-VATSAL JASANI");

}

public static void array\_count9(int h[],int j)

{

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

{

if(h[i]==9)

count++;

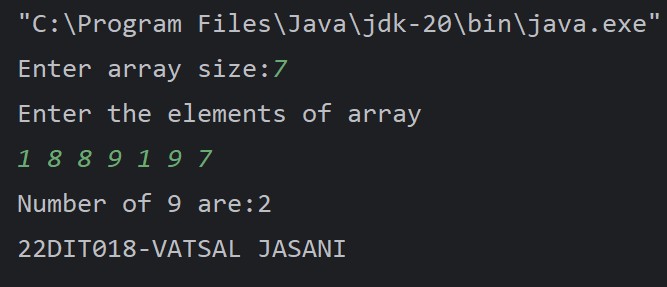
}

System.*out*.println("Number of 9 are:"+count);

}

}

**Output :-**



## Conclusion :-

We use the concept of if…else to print the number of repeated element.

## [9.] Aim :-

Given a string, return a string where for every char in the original, there are two chars. double\_char('The') → 'TThhee' double\_char('AAbb') → 'AAAAbbbb' double\_char('Hi-There') → 'HHii--TThheerree'

## Program code :-

import java.util.Scanner; class prac {

public static void main(String[] args)

{

System.*out*.println("enter a string: "); Scanner sc=new Scanner (System.*in*); String a=sc.nextLine();

int l=a.length(); for(int i=0;i<l;i++)

{

System.*out*.print(a.charAt(i)); System.*out*.print(a.charAt(i));

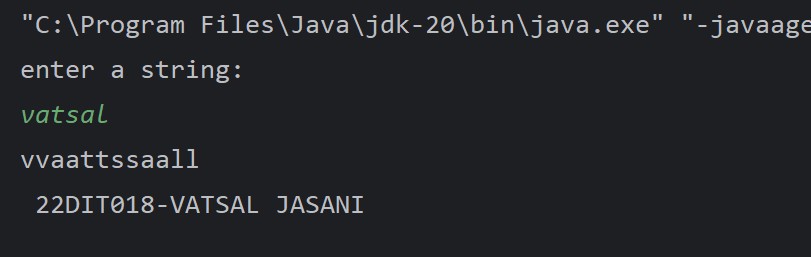
}

System.*out*.println("\n 22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

We learnt about how to use the library function ‘charAt’ and we also learned about how it works**.**

## [10.] Aim :-

Perform following functionalities of the string: ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String ● Sort the string.

## Program code :-

import java.util.Scanner; import java.util.Arrays; public class prac

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.*in*); System.*out*.println("enter a string:"); String a=sc.next();

int l=a.length();

StringBuilder str= new StringBuilder(a); System.*out*.println("string length is:"+l); System.*out*.println("into lowercase:"+a.toLowerCase()); System.*out*.println("into uppercase:"+a.toUpperCase()); System.*out*.println("your string is:"+str.reverse()); char h[]=a.toCharArray();

Arrays.*sort*(h); for(int i=0;i<l;i++)

{

System.*out*.print(h[i]+ " ");

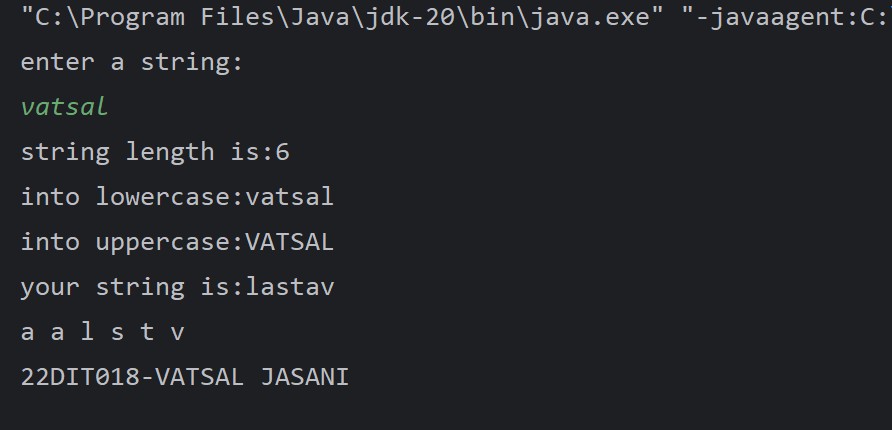
}

System.*out*.println("\n22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Here we learnt about various function of String library.Also we learned about the ‘toUpperCase’ and ‘toLowerCase’function.This function name itself suggest that we can alter a strings case to upper or lower case by using this functions.

## [11.] Aim :-

Perform following Functionalities of the string: “CHARUSAT UNIVERSITY”

* Find length ● Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’ ● Convert all character in lowercase

## Program code :-

import java.util.Scanner; public class prac

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.*in*); System.*out*.println("enter a string:"); String a=sc.next();

int l=a.length(); System.*out*.println("string length is:"+l);

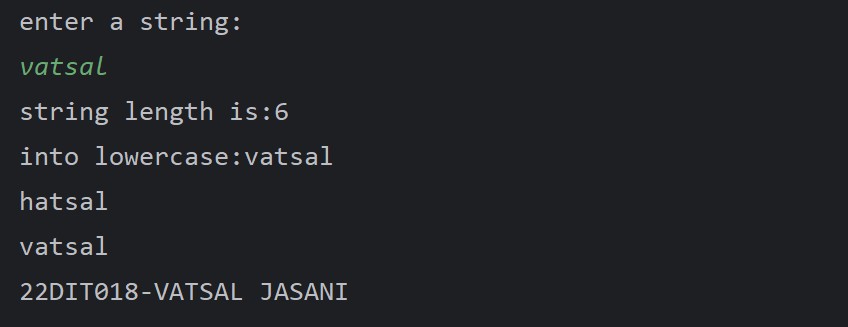
System.*out*.println("into lowercase:"+a.toLowerCase()); System.*out*.println(a.replace('v', 'h')); System.*out*.println(a);

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Here we learnt about various function of String library. We learned about ‘replace’ function which replaces the string with the desired string.Also we learned about the ‘toUpperCase’ and ‘toLowerCase’function.This function name itself suggest that we can alter a strings case to upper or lower case by using this functions

**PART-III**

# Object Oriented Programming: Classes, Methods, Constructors [12.] Aim :-

Write a java program for converting Pound into Rupees. (Accept Pounds from command line argument and using scanner class also and take 1 Pound = 100 Rupees.)

**Program code :-**

import java.util.\*; public class prac {

public static void main(String[] args)

{

Scanner s = new Scanner(System.*in*); int pound;

int rupee=0; System.*out*.print("Add pound: "); pound = s.nextInt();

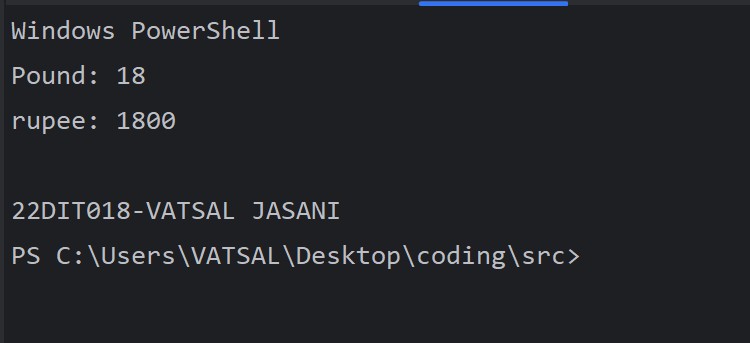
rupee = pound\*100; System.*out*.println("Pound: "+pound); System.*out*.println("rupee: "+rupee);

System.*out*.println("\n22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input.

# [13.] Aim :-

Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10% raise and display each Employee’s yearly salary again.

**Program code :-**

import java.util.Scanner; class employee

{

double salary; String fn; String ln; employee()

{

fn=null; ln=null; if(salary<0)

{

salary=0.00;

}

}

void get()

{

Scanner sc=new Scanner(System.*in*); System.*out*.println("enter your your first name:"); fn=sc.nextLine();

System.*out*.println("enter your last name:"); ln=sc.nextLine();

System.*out*.println("enter a monthly salary"); salary=sc.nextDouble();

}

void put()

{

System.*out*.println("your name is:" +fn+ " " +ln); System.*out*.println("your monthly salary is:" +salary);

}

}

class prac

{

public static void main(String[] args)

{

employee e1=new employee(); System.*out*.println("for employee 1 ");

e1.get();

employee e2=new employee(); System.*out*.println("for employee 2 ");

e2.get(); System.*out*.println();

System.*out*.println("for employee 1 ");

e1.put();

System.*out*.println("for employee 2 ");

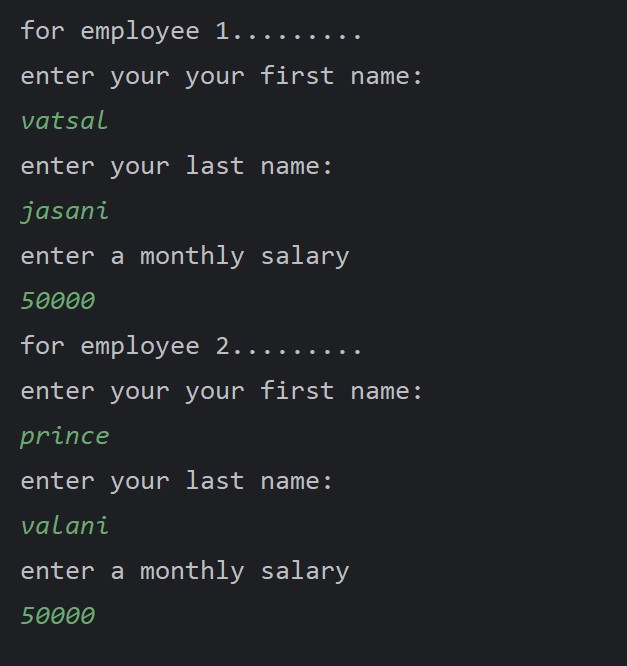
e2.put(); e1.salary=e1.salary+(e1.salary\*0.10); e2.salary=e2.salary+(e2.salary\*0.10);

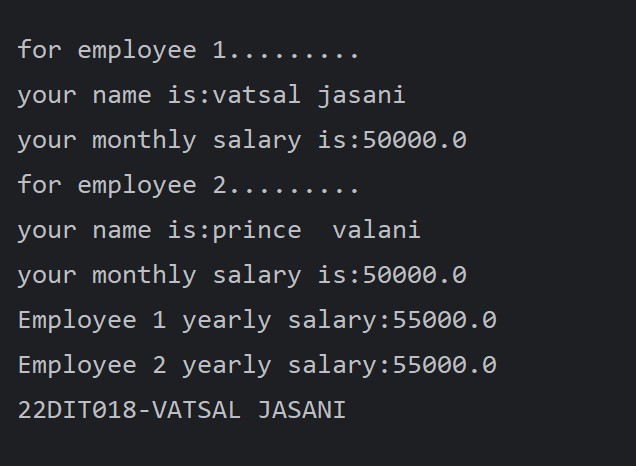
System.*out*.println("Employee 1 yearly salary:" +e1.salary); System.*out*.println("Employee 2 yearly salary:" +e2.salary); System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**





## Conclusion :-

Here we learnt about how to [classes](https://www.geeksforgeeks.org/understanding-classes-and-objects-in-java/) consist of both [constructors](https://www.geeksforgeeks.org/constructors-in-java/) and [methods](https://www.geeksforgeeks.org/methods-in-java/) and how to use in main method.

# [14.]Aim :-

Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date’s capabilities.

**Program code :-**

import java.util.Scanner; class Date

{

int m,d,y; Date()

{

m=0; d=0; y=0;

}

void get()

{

Scanner sc=new Scanner(System.*in*); System.*out*.println("enter a month:"); m=sc.nextInt(); System.*out*.println("enter a day:"); d=sc.nextInt(); System.*out*.println("enter a year:"); y=sc.nextInt();

}

void put()

{

System.*out*.println("Date:"+d+"/"+m+"/"+y);

}

}

class prac

{

public static void main(String[] args)

{

Date d1=new Date(); d1.get();

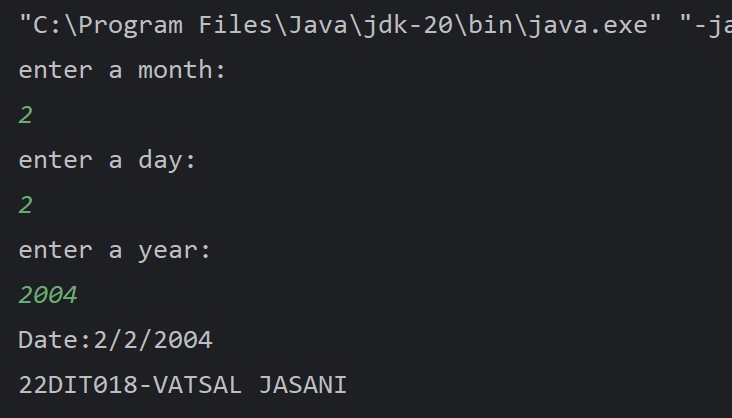
d1.put();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Here we learnt about how to [classes](https://www.geeksforgeeks.org/understanding-classes-and-objects-in-java/) consist of both default [constructors](https://www.geeksforgeeks.org/constructors-in-java/) and [methods](https://www.geeksforgeeks.org/methods-in-java/).

# [15.] Aim :-

Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle.

Length and breadth of rectangle are entered through keyboard

**Program code :-**

import java.util.\*; class prac

{

double length; double breadth;

prac(double l,double b)

{

length=l; breadth=b;

}

void returnArea()

{

double area;

area= length\*breadth;

System.*out*.println("The area of rectangle is:"+area);

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.*in*); System.*out*.println("enter a length:"); float l=sc.nextInt(); System.*out*.println("enter a breadth:"); float b=sc.nextInt();

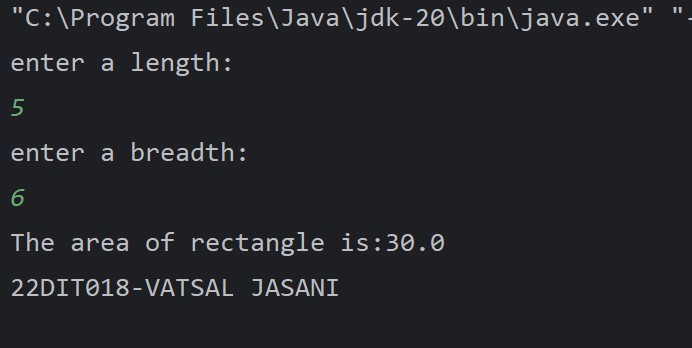
prac a1=new prac(l,b); a1.returnArea();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



**Conclusion :-**

Parameterized constructors are used to create user instances of objects with user defined states. There can be more than one parameterized constructor in a class.

# [16.] Aim :-

Print the sum, difference and product of two complex numbers by creating a class named ‘Complex’ with separate methods for each operation whose real and imaginary parts are entered by user.

**Program code :-**

import java.util.Scanner; public class prac {

int real, imag;

prac() {

real = 0;

imag = 0;

}

void sum(prac c) {

int ansr = real + c.real; int ansi = imag + c.imag;

System.*out*.println("ADDITION: " + ansr + " + " + ansi +

"i");

}

void diff(prac c) {

int ansr = real - c.real; int ansi = imag - c.imag;

if (ansi >= 0)

System.*out*.println("SUBTRACTION: " + ansr + " + " + ansi

+ "i");

else

System.*out*.println("SUBTRACTION: " + ansr + " " + ansi +

"i");

}

void mul(prac c) {

int ansr = real \* c.real; int ansi = imag \* c.imag;

System.*out*.println("MULTIPLICATION: " + ansr + " + " + ansi

+ "i");

}

public static void main(String args[]) { Scanner sc = new Scanner(System.*in*);

prac c1 = new prac();

System.*out*.println("\nEnter first real part: "); c1.real = sc.nextInt();

System.*out*.println("Enter first imaginary part: "); c1.imag = sc.nextInt();

prac c2 = new prac(); System.*out*.println("\nEnter second real part: "); c2.real = sc.nextInt();

System.*out*.println("Enter second imaginary part: "); c2.imag = sc.nextInt();

c1.sum(c2);

c1.diff(c2);

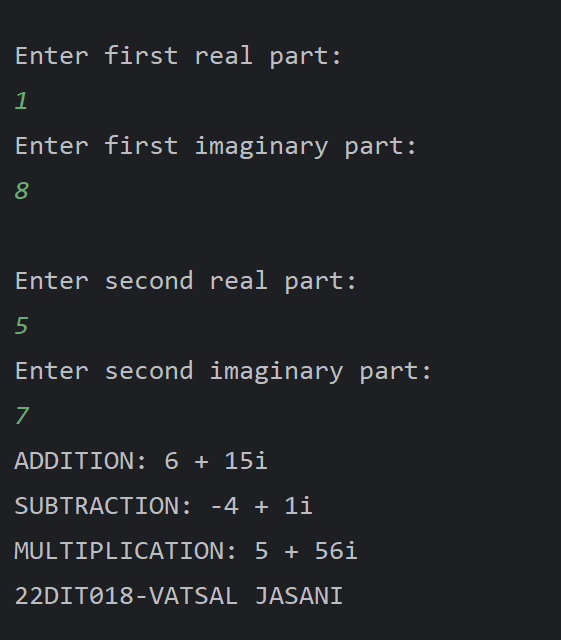
c1.mul(c2);

System.*out*.println("22DIT018-VATSAL JASANI\n");

}

}

**Output :-**



## Conclusion :-

the parameterized method takes a parameter or a list of the parameters that receive the respective values from the method calling.

**PART-IV**

# Aim :-

## Inheritance, Interface, Package

Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent 1 class 2

- method of child class by object of child class 3 - method of parent class by object of child class.

## Program code :-

class pr17 {

void Display(){

System.*out*.println("this is parent class");

}

}

class pr172 extends pr17{ void print(){

System.*out*.println("this is child class");

}

}

class prac{

public static void main(String[] args){ pr17 p=new pr17();

p.Display();

pr172 p1=new pr172(); p1.print();

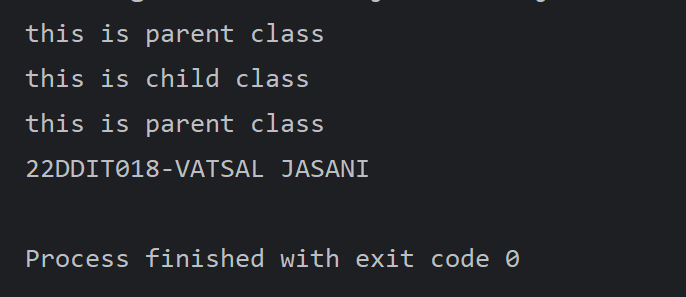
p1.Display();

System.*out*.println("22DDIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

The inheritance in which a single derived class is inherited from a single base class is known as the Single Inheritance.

1. **Aim :-**

Create a class named 'Member' having the following members: Data members 1

- Name 2 - Age 3 - Phone number 4 - Address 5 – Salary It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

## Program code :-

import java.util.Scanner; class Member {

String Name; int age; double ph; String add;

float salary=2000000; void printsalary(){

System.*out*.println("salary:"+salary);

}

}

class manager extends Member{ String spec;

String dep;

}

class emp extends Member{ String spec;

String dep;

}

class prac

{

public static void main(String[] args){ manager m=new manager();

emp e = new emp(); e.Name="vatsal"; e.age=18; e.ph=586142381;

e.add="surat"; e.spec="data-scientist"; e.dep="IT";

System.*out*.println("for manager ");

System.*out*.println("name is:"+e.Name); System.*out*.println("age is:"+e.age);

System.*out*.println("phone number is:"+e.ph); System.*out*.println("address is:"+e.add); System.*out*.println("specialization is:"+e.spec); System.*out*.println("department is:"+e.dep); e.printsalary();

System.*out*.println("\n- -

");

m.Name="Prince"; m.age=18; m.ph=1593348525; m.add="NADIAD"; m.spec="WEB TECH.";

m.dep="CSE";

System.*out*.println("for Employee ");

System.*out*.println("name is:"+m.Name); System.*out*.println("age is:"+m.age); System.*out*.println("phone number is:"+m.ph); System.*out*.println("address is:"+m.add); System.*out*.println("specialization is:"+m.spec); System.*out*.println("department is:"+m.dep); m.printsalary();

System.*out*.println("................................................

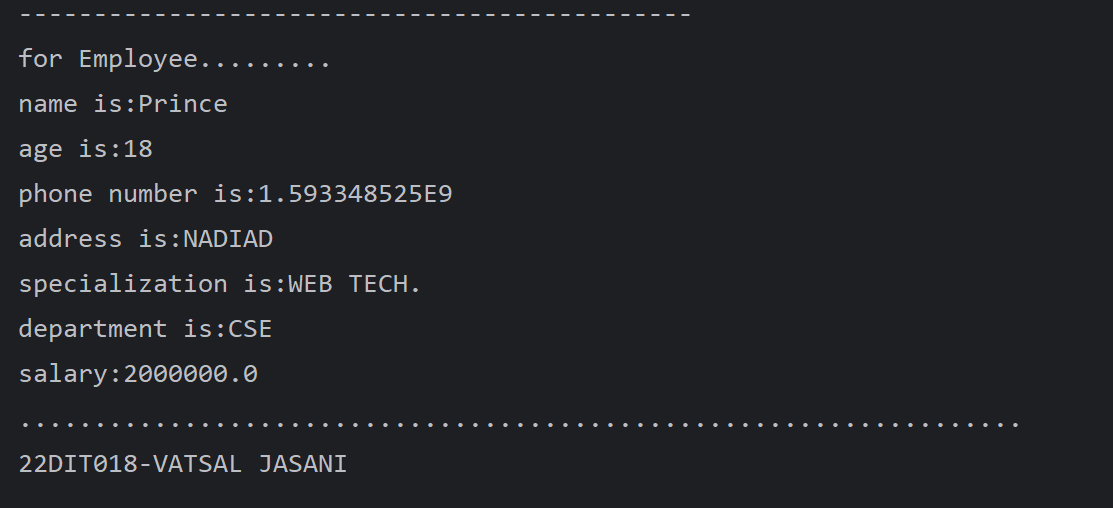
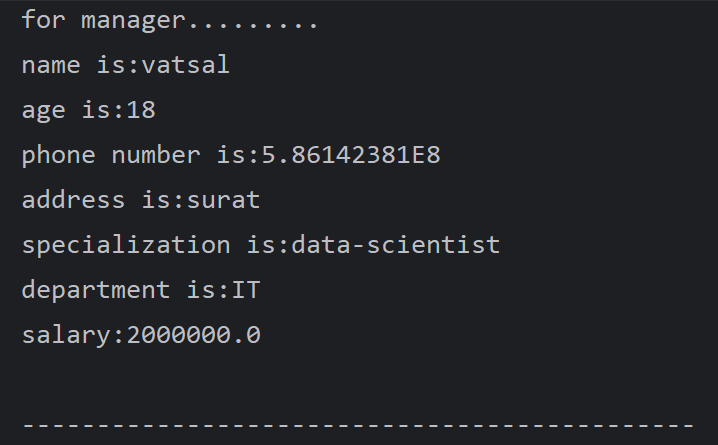
...................");

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

## Output :-



**Conclusion :-**

Hierarchical Inheritance in Java is a type of inheritance in Java where a single parent class can have multiple child classes.

# Aim :-

Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.

**Program code :-**

class rectangle{ int length; int breadth;

rectangle(int l,int b)

{

length=l; breadth=b;

}

void area()

{

int area; area=length\*breadth;

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

}

void per()

{

int pr; pr=2\*(length+breadth);

System.*out*.println("Perimeter is:"+pr);

}

}

class square extends rectangle{ square(int s)

{

super(s,s);

}

}

class prac {

public static void main(String[] args)

{

rectangle r=new rectangle(8,12); square sq=new square(15);

System.*out*.println("For rectangle ");

r.area();

r.per();

System.*out*.println("For square ");

sq.area();

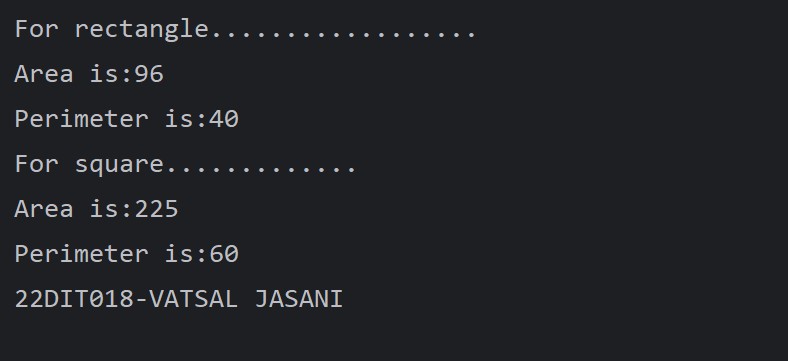
sq.per();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

The super keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor.

# Aim :-

Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

**Program code :-**

class shape {

void print(){

System.*out*.println("this is shape");

}

}

class Rectangle1 extends shape{ void print1(){

System.*out*.println("This is rectangular shape");

}

}

class Circle extends shape{ void print2(){

System.*out*.println("This is circular shape");

}

}

class squre extends Rectangle1{ void print3(){

System.*out*.println("Square is a rectangle.");

}

}

class D{

public static void main(String[] args) { squre s=new squre();

s.print();

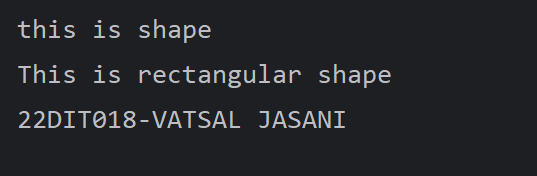
s.print1();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

* + In Multi-Level Inheritance in Java, a class extends to another class that is already extended from another class**.**
  + Hierarchical Inheritance in Java is a type of inheritance in Java where a single parent class can have multiple child classes.

# Aim :-

Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

**Program code :-**

class degree {

void getdegree(){

System.*out*.println("I got a Degree");

}

}

class Undergraduate extends degree{ void getdegree(){

System.*out*.println("I am an Undergraduate");

}

}

class Postgraduate extends degree{ void getdegree(){

System.*out*.println("I am an Postgraduate");

}

}

class deg{

public static void main(String[] args) { degree d1=new degree(); Postgraduate p1=new Postgraduate();

Undergraduate U1=new Undergraduate(); d1.getdegree();

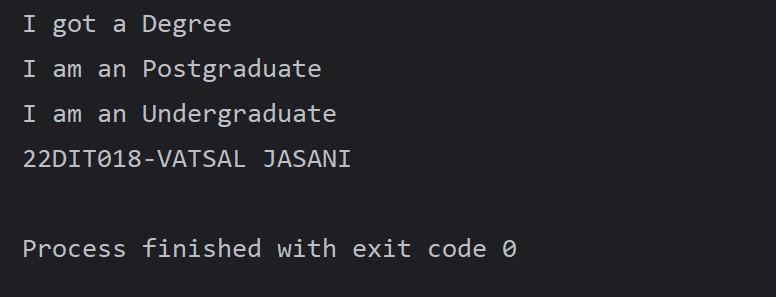
p1.getdegree(); U1.getdegree();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

* + In Multi-Level Inheritance in Java, a class extends to another class that is already extended from another class**.**

# Aim :-

Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor\_sum(int n). You need to write a class calledMyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example, divisors of 6 are 1, 2, 3 and 6, so divisor\_sum should return 12. The value of n will be at most 1000.

**Program code :-**

import java.util.Scanner;

interface AdvanceArithmatic{ int divisor\_sum(int n);

}

class MyCalculator implements AdvanceArithmatic{ public int divisor\_sum(int n){

int sum=n;

for(int i=1;i<=(n/2);i++)

{

if(n%i==0)

sum+=i;

}

return sum;

}

}

class pr22

{

public static void main(String[] args) { AdvanceArithmatic a1=new MyCalculator(); Scanner sc=new Scanner(System.*in*);

int h; while(true)

{

System.*out*.println("enter a value:"); h=sc.nextInt();

if(h<0 || h>1000){

System.*out*.println("error:404 \n Please,enter a value between th value of 0 and 1000 ");

}

else{break;}

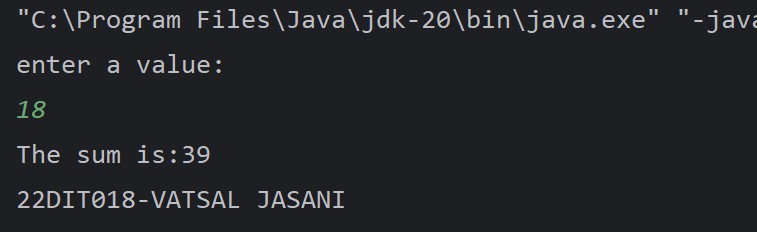
}

int sum=a1.divisor\_sum(h); System.*out*.println("The sum is:" +sum) ; System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Interfaces in Java help achieve Polymorphism and Abstraction. They also make it possible to implement Multiple Inheritance in Java.

# Aim :-

Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.

**Program code :-**

import java.util.Scanner; interface shapes{

void area(); void Perimeter();

default void color(){ System.*out*.println("Color is:Navy Blue");

}

}

class rectangle1 implements shapes{ int l,b;

rectangle1(int length,int width)

{

l=length; b=width;

}

public void area(){ int area=l\*b;

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

}

public void Perimeter(){ int per=2\*(l+b);

System.*out*.println("Perimeter is:"+per);

}

public void display(){ area(); Perimeter(); color();

}

}

class circle implements shapes{ int a;

circle(int s)

{

a=s;

}

public void area(){

float area=(float)(3.14\*a\*a); System.*out*.println("Area is:"+area);

}

public void Perimeter(){

float per=(float)(2\*3.14\*a); System.*out*.println("Perimeter is:"+per);

}

public void display(){ area(); Perimeter(); color();

}

}

class shape{

public static void main(String[] args) { Scanner sc=new Scanner(System.*in*);

System.*out*.println("for rectengle ");

System.*out*.print("Enter a length:"); int l=sc.nextInt(); System.*out*.print("Enter a width:"); int b=sc.nextInt();

rectangle1 r=new rectangle1(l, b); r.display();

System.*out*.println("For circle ");

System.*out*.print("enter radius:"); int h=sc.nextInt();

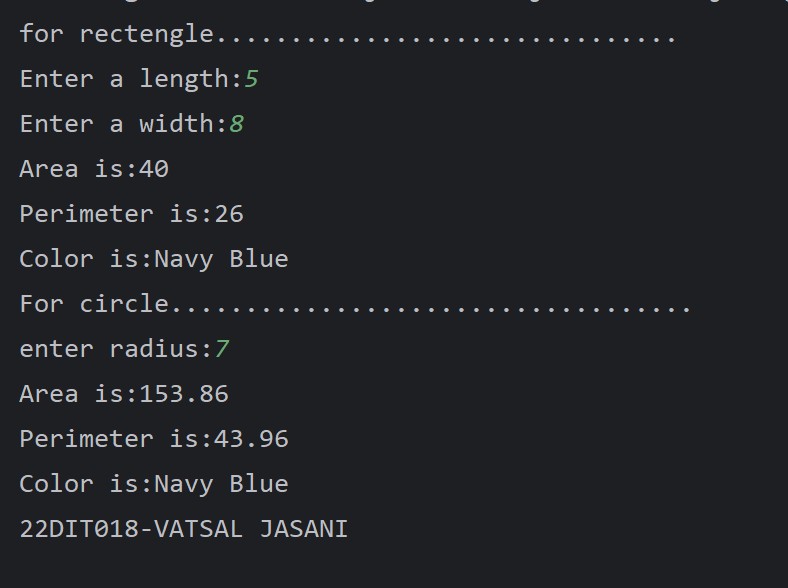
circle c=new circle(h); c.display();

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Default methods enable you to add new functionality to existing interfaces and ensure binary compatibility with code written for older versions of those interfaces.

**PART-V**

# Exception Handling

**[24.] Aim :-**

Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.

**Program code :-**

import java.util.Scanner; class pr24 {

public static void main(String[] args) { Scanner s=new Scanner(System.*in*); System.*out*.print("Enter a value of x:"); int x=s.nextInt(); System.*out*.print("Enter a value of y:"); int y=s.nextInt();

try {

int z=x/y; System.*out*.println("value is:"+z);

} catch (Exception e) { System.*out*.println(e);

System.*out*.println("please!check the value of x or y.");

}

finally{

System.*out*.println("DONE ");

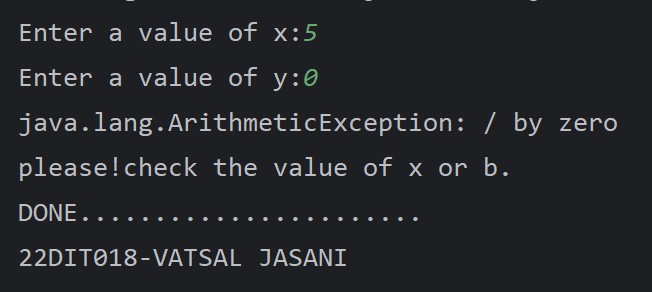
System.*out*.println("22DIT018-VATSAL JASANI");

}

}

}

**Output :-**



## Conclusion :-

* The try block is used to specify a block of code that may throw an exception.
* The catch block is used to handle the exception if it is thrown.
* The finally block is used to execute the code after the try and catch blocks have been executed.

# [25.] Aim :-

Write a Java program that throws an exception and catch it using a try-catch block.

**Program code :-**

import java.util.Scanner; public class pr25 {

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

Scanner sc= new Scanner(System.*in*); System.*out*.print("Enter a value of n:"); int n=sc.nextInt();

int a[]=new int[n]; for(int i=0;i<n;i++){

a[i]=sc.nextInt();

}

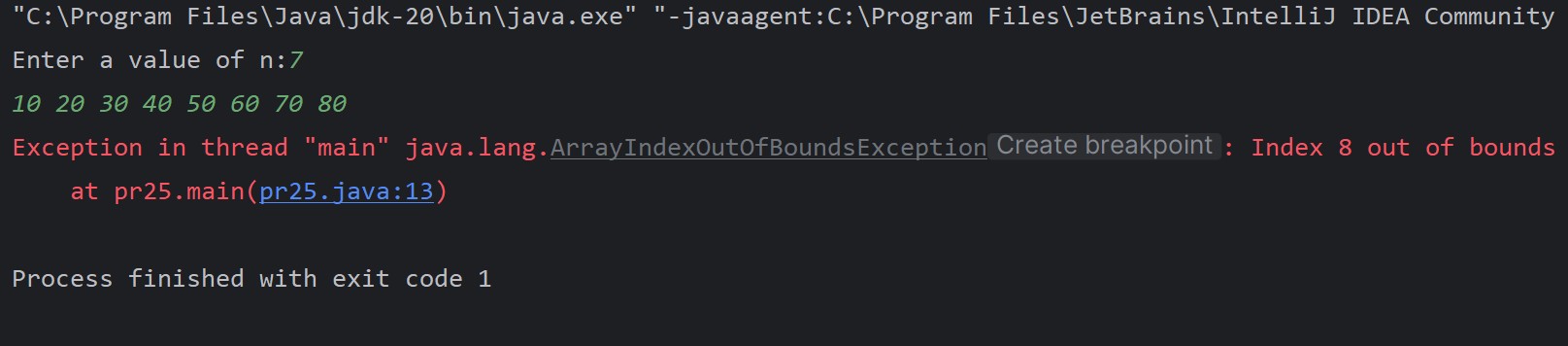
int h=sc.nextInt(); h=a[n+1];

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



## Conclusion :-

Throws keyword is used for exception handling in java, where one needs to handle the flow of the program when a checked exception occurs.

# [26.] Aim :-

Write a java program to generate user defined exception using “throw” and “throws” keyword.Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exception in program).

**Program code :-**

import java.io.IOException;

class MyCustomException extends Exception { public MyCustomException(String message) {

super(message);

}

}

public class pr25 {

public static void checkedExceptionExample() throws IOException

{

throw new IOException("This is a checked exception");

}

public static void uncheckedExceptionExample() { throw new RuntimeException("This is an unchecked

exception");

}

public static void customExceptionExample(int value) throws MyCustomException {

if (value < 0) {

throw new MyCustomException("Negative value is not

allowed");

}

}

public static void main(String[] args) { try {

*checkedExceptionExample*();

} catch (IOException e) {

System.*out*.println("Caught Checked Exception: " + e.getMessage());

}

try {

*uncheckedExceptionExample*();

} catch (RuntimeException e) { System.*out*.println("Caught Unchecked Exception: " +

e.getMessage());

}

try {

*customExceptionExample*(-5);

} catch (MyCustomException e) { System.*out*.println("Caught Custom Exception: " +

e.getMessage());

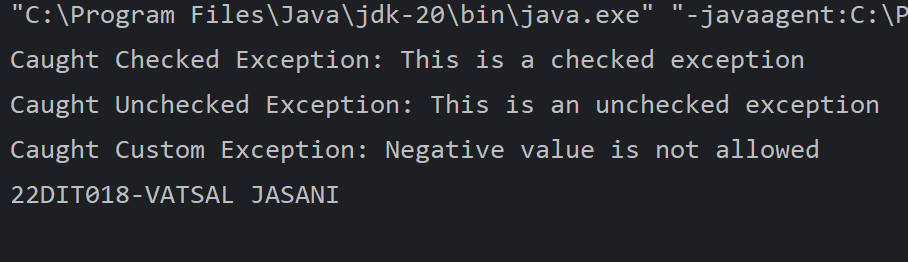
}

System.*out*.println("22DIT018-VATSAL JASANI");

}

}

**Output :-**



**Conclusion :-**

The Throw keyword throws an exception explicitly. Whereas the Throws keyword declares that a method might throw an exception.