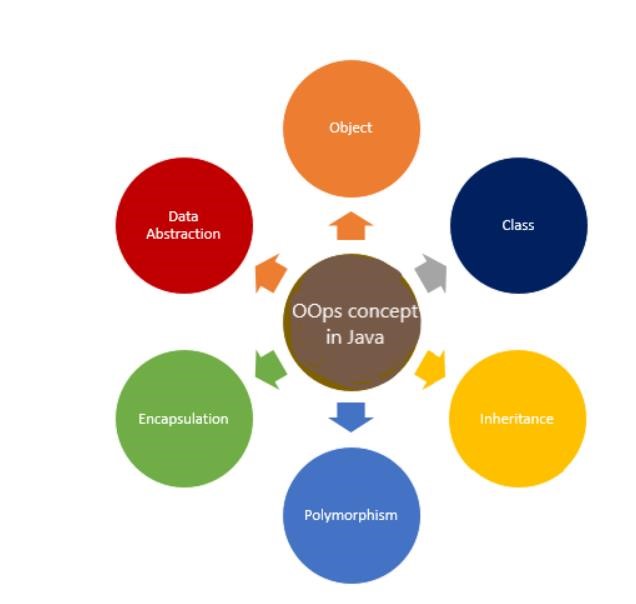
**Q1) Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM.**

i) Introduction to Object Oriented Concepts

➢ OOps in java is to improve code readability and reusability by defining a Java program efficiently. The main principles of object-oriented programming are **abstraction, encapsulation, inheritance, and polymorphism**. These concepts aim to implement real-world entities in programs.



## ❖ Class :-

* It is a user-defined data type, which holds its own **data members and member functions**, which can be accessed and used by creating an instance of that class.
* A class is like a **blueprint for an object**. Data members => data variables and member functions => functions
* Functions are used to manipulate these variables and together these data members and member functions define **the properties and behavior** of the objects in a Class.

### Example:-

Class of Cars . There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc will act as data members. Applying brakes, increasing speed may lead to member function of that class.

## ❖ Object:-

* Objects are the runtime entities, existing in the real world. They can be represented as person, a place, a bank account.
* Real-world objects have attributes and behaviors.

**Examples:**

Dog =>

Attributes: breed, color, hungry, tired, etc.

Behaviors: eating, sleeping, etc.

Bank Account =>

Attributes: account number, owner, balance

Behaviors: withdraw, deposit

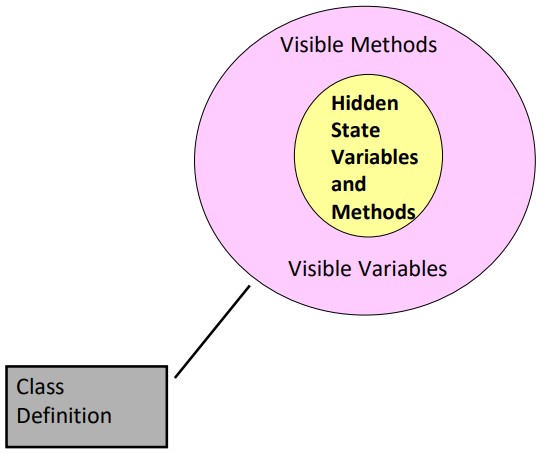
### Note:-

When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

On class can have multiple objects, which can interact with each other

## ❖ Encapsulation:-

* he wrapping up of data and functions into a single unit is known as encapsulation.
* When classes are defined, programmers can specify that certain methods or state variables remain hidden inside the class.
* These variables and methods are accessible from within the class, but not accessible outside it.
* The insulation of the data from direct access by the program is called data hiding or information hiding.



## ❖ Abstraction:-

* Abstraction refers to the act of representing essential features without including the background details or explanations.
* Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and functions to operate on these attributes. They encapsulate all the essential properties of the objects that are to be created.
* Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc. in the car. This is what abstraction is.

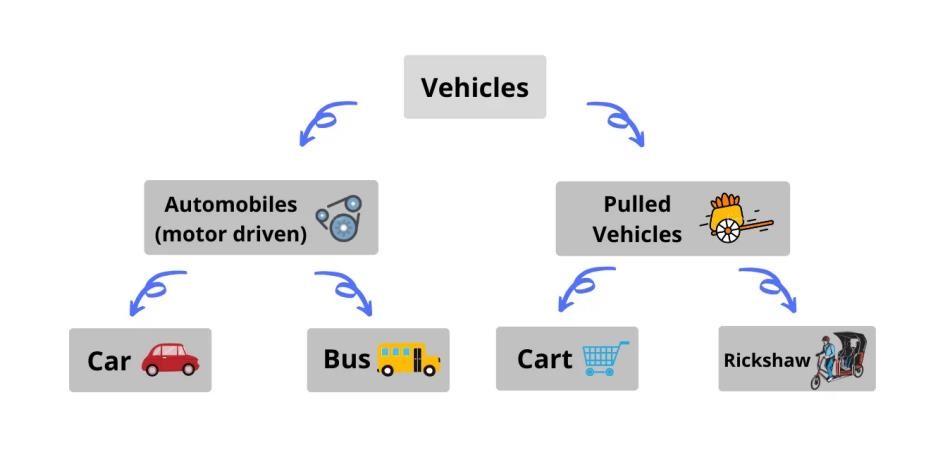
## ❖ Inheritance:-

* Inheritance is the process by which objects of one class acquire the properties of objects of another class.
* The capability of a class to derive properties and characteristics from another class is called Inheritance.
* Inheritance is one of the most important features of Object-Oriented Programming.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

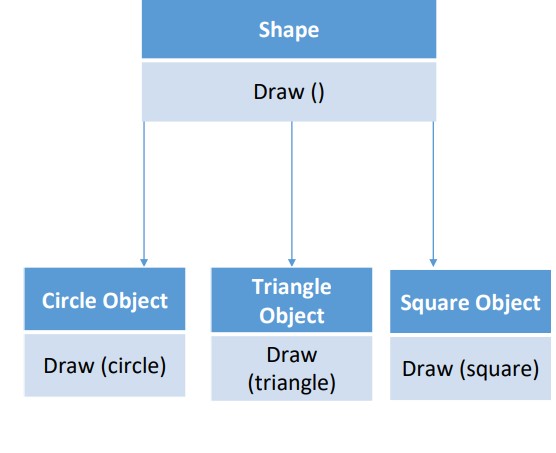
**Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

**Reusability**: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.



## ❖ Polymorphism:-

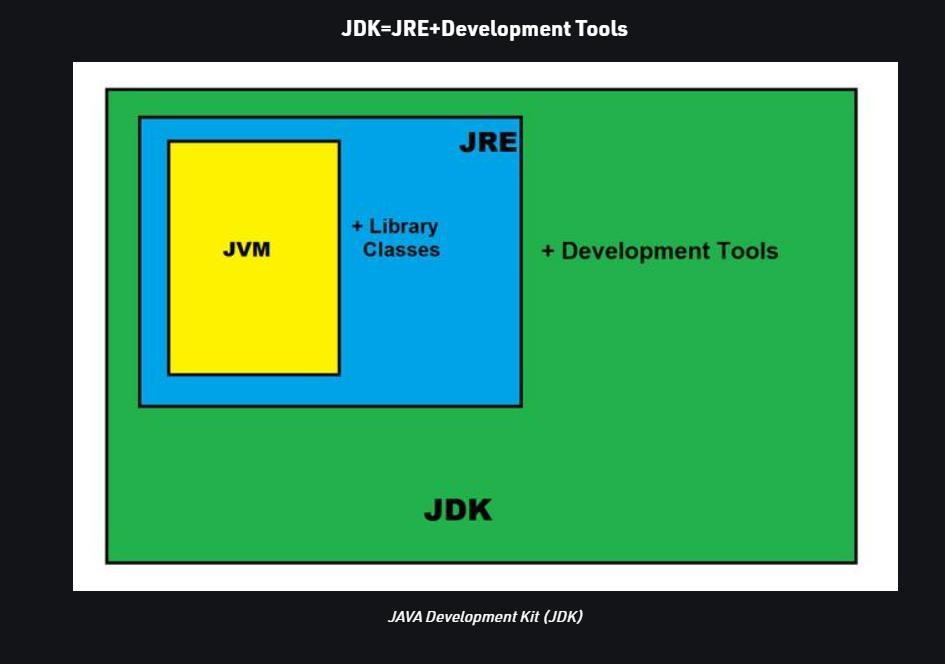
* Polymorphism is one of the essential features of an object- oriented language. It means ability to take more than one form.
* An operation may exhibit different behaviours in different instances. The behavior depends upon the type of data used in the operations.
* Example: Addition operation, with integer datatype, it will generate sum, but if operands are strings, it will generate third string by concatenation.
* The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.
* Single function name can be used to handle different number and different types of arguments.
* This is something similar to a particular word having several different meaning depending on the context. So, using a single function name to perform different types of tasks is known as function overloading.
* Polymorphism is extensively used in implementing inheritance.



### *ii) comparison of Java with c++:-*

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Java** | **C++** |
| Founder | Java was developed by James Gosling at Sun  Microsystems. | C++ was developed by  Bjarne Stroustrup at Bell Labs in 1979 as an extension of the C language. |
| First Release | On May 23, 1995 | In October 1985 |
| Stable Release | Java SE 14 or JDK 14 was released on March 17, 2020. | C++17 was released in December 2017. |
| Official Website | oracle.com/java | isocpp.org |
| Influenced By: | Java was Influenced by Ada 83, Pascal, C++, C#, etc. languages. | C++ was Influenced by  Influenced by Ada, ALGOL 68, C, ML, Simula, Smalltalk, etc. languages. |
| Influenced to: | Java was influenced to develop BeanShell, C#, Clojure, Groovy, Hack, J#, Kotlin, PHP, Python, Scala, etc. languages. | C++ was influenced to develop C99, Java, JS++, Lua, Perl, PHP, Python, Rust, Seed7, etc. languages. |
| Platform Dependency | Platform independent, Java bytecode works on any operating system. | Platform dependent, should be compiled for different platforms. |
| Portability | It can run in any OS hence it is portable. | C++ is platform-dependent. Hence it is not portable. |
| Compilation | Java is both Compiled and Interpreted Language. | C++ is a Compiled Language. |
| Memory Management | Memory Management is System Controlled. | Memory Management in C++ is Manual. |
| Virtual Keyword | It doesn’t have Virtual  Keyword. | It has Virtual keywords. |
| Multiple Inheritance | IIt supports only single inheritance. Multiple inheritances are achieved partially using interfaces. | It supports both single and multiple Inheritance. |
| Overloading | It supports only method overloading and doesn’t allow operator overloading. | It supports both method and operator overloading. |
| Pointers | It has limited support for pointers. | It strongly supports pointers. |
| Libraries | It doesn’t support direct native library calls but only Java Native Interfaces. | It supports direct system library calls, making it suitable for system-level programming. |
| Libraries | Libraries have a wide range  of classes for various highlevel services. | C++ libraries have comparatively low-level functionalities. |
| Documentation Comment | It supports documentation  comments (e.g., /\*\*.. \*/) for source code. | It doesn’t support documentation comments for source code. |
| Thread Support | Java provides built-in support for multithreading. | C++ doesn’t have built-in support for threads, depends on third-party threading libraries. |
| Type | Java is only an objectoriented programming language. | C++ is both a procedural and an object-oriented programming language. |
| Input-Output mechanism | Java uses the (System class): System.in for input and System.out for output. | C++ uses cin for input and cout for an output operation. |
| goto Keyword | Java doesn’t support goto  Keyword | C++ supports goto keyword. |
| Structures and Unions | Java doesn’t support Structures and Unions. | C++ supports Structures and Unions. |
| Parameter Passing | Java supports only the Pass by Value technique. | C++ supports both Pass by Value and pass by reference. |
| Global Scope | It supports no global scope. | It supports both global scope and namespace scope. |
| Object Management | Automatic object management with garbage collection. | It supports manual object management using new and delete. |
| Call by Value and Call by  reference | Java supports only call by value. | C++ both supports call by value and call by reference. |
| Hardware | Java is not so interactive with hardware. | C++ is nearer to hardware. |

*iii) Introduction to JDK, JRE, JVM:-*



❖ *JDK:-*

* JDK contains everything that JRE has alongside improvement devices for creating, troubleshooting, and observing Java applications. You require JDK when you have to build Java applications.

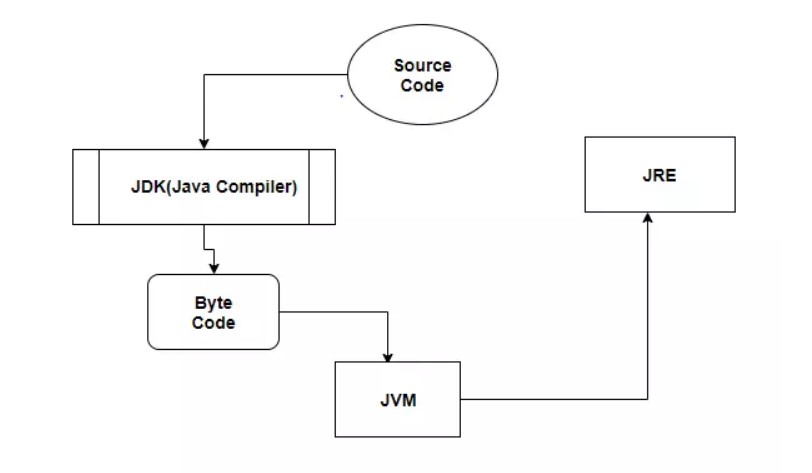
* The Java Development Kit (JDK) provides an environment utilized for creating Java applications and applets. It incorporates the Java

Runtime Environment (JRE), a loader, a compiler (javac), an

archiver (jar), a documentation generator (Javadoc) and different apparatuses required in Java improvement.

# ❖ JRE:-

* The Java Runtime Environment (JRE) is a product bundle which packages the libraries (containers) and the Java Virtual Machine, and different parts to run applications written in Java. JVM is only a piece of JRE dispersions.
* To execute any Java application, you require JRE introduced in the machine. It’s base necessity to run Java applications on any device.
* JREs come as a feature of JDKs, or you can download them independently. JREs are dependent on the kind of machine (OS and engineering). You should choose the JRE package to import and introduce.



# ❖ JVM:-

* It is also known as Java Virtual Machine. It acts as a run-time motor to run Java applications. JVM is the one that calls the entry-point “main” method. JVM is a piece of JRE(Java Runtime Environment).
* An engineer can create Java code on one framework and can anticipate that it should keep running on some other Java empowered framework with no modification. It is all conceivable due to JVM.
* When we assemble a .java document, the Java compiler produces the .class files with similar names present in the .java record. This .class document goes into different advances when we run it. These means together depict the entire JVM.
* JVM conveys the ideal execution for Java applications utilizing many propelled strategies, fusing a best in class memory display/model, garbage collector, and versatile, adaptive optimizer.
* The server VM has been uncommonly tuned to amplify top working rate. It has to run for long-running server applications, which requires the quickest conceivable and active velocity over quick start-up time. Engineers can pick which framework they need by determining – customer or – server.
* The JVM is virtual because it gives a machine interface that does not rely upon the basic working framework and machine equipment design. This freedom from equipment and the working structure is a foundation of the write once-run-anyplace estimation of Java programs.

**Compare JVM, JDK, and JRE**

* JVM is the virtual engine and the one which enables byte code support.
* JRE contains JVM and all the other libraries to run Java application. It is enough to run any Java application.
* JDK is a superset which comprises of JVM, JRE, and the tools to develop Java Application.

Q 2

import java.util.\*;

class q2{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        String s = sc.next();

        char a = s.charAt(0);

        char b = s.charAt(1);

        if(s.charAt(0)=='o'||s.charAt(0)=='O')

        {

            if(s.charAt(1)=='z'||s.charAt(0)=='Z')

            {

                System.out.println(a+""+b);

            }

            else

            System.out.println(a);

        }

        else

        {

            if(s.charAt(1)=='z'||s.charAt(0)=='Z')

            {

                System.out.println(b);

            }

        }

    }

}

Q3

import java.util.\*;

public class q3 {

    static boolean lastDigit(int a,int b){

        if(a%10 == 0 && b% 10 == 0)

        {

            return true;

        }

        else if(b % 10 ==a || a % 10 == b || a ==b ){

            return true;

        }

        else

        return false ;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 1st no");

        int a = sc.nextInt();

        System.out.println("Enter 2nd no");

        int b = sc.nextInt();

        System.out.println(lastDigit(a,b));

    }

}

Q4

import java.util.\*;

public class q4 {

    static public boolean arr123(int[] arr) {

        boolean flag = false;

        for (int i = 0; i < arr.length - 2; i++) {

            // if (arr[i] != 1 && arr[i + 1] != 2 && arr[i + 2] != 3) {

            //     continue;

            // }

            if (arr[i] == 1 && arr[i + 1] == 2 && arr[i + 2] == 3) {

                flag = true;

            }

        }

        return flag;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        System.out.println("Enter length of array: ");

        int n = sc.nextInt();

        int arr[] = new int[n];

        System.out.println("Enter the array");

        for (int i = 0; i < n; i++)

        arr[i] = sc.nextInt();

        boolean flag = arr123(arr);

        System.out.println(flag);

        // Appending new elements at

        // the end of the list

        // String s = arr.toString();

        // if(s.contains("123"))

        // {

        // System.out.println("true");

        // }

        // for (int i = 0; i < n - 2; i++) {

        // if (arr[i] != 1 && arr[i + 1] != 2 && arr[i + 2] != 3) {

        // System.out.println("false");

        // //break;

        // }

        // else if(arr[i] == 1 && arr[i + 1] == 2 && arr[i + 2] == 3)

        // {System.out.println("true");

        // break;

        // }

        // }

    }

}

Q5

import java.util.\*;

public class q5 {

    static int stringMatch(String a,String b)

{

    int counter = 0;

    for(int i=0;(i<a.length()-2) && (i<b.length()-2); i++)

    {

        String asub = a.substring(i,i+2);

        String bsub = b.substring(i,i+2);

        if(asub.equals(bsub)){

            counter ++;

        }

    }

    return counter;

}

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 1st string:");

        String a = sc.next();

        System.out.println("Enter 2nd string:");

        String b = sc.next();

        System.out.println("No of matching substrings"+stringMatch(a, b));

    }

}

Q6

import java.util.Scanner;

public class q6 {

    public static void main(String[] args) {

        System.out.println("Enter length of string array");

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        String []arr = new String[n];

        for(int i = 0 ;i<n ;i++)

        {

            arr[i] = sc.next();

        }

        System.out.println("Enter string for removal");

        String rm = sc.next();

        int j=0;

        for( int i = 0 ;i<n ;i++)

        {

            if(!arr[i].equals(rm))

            {

                j++;

            }

        }

        String []arr1 = new String[j];

        for( int i = 0 ;i<j ;i++)

        {

            if(!arr[i].equals(rm))  // cnt use == and != because they are objects and there is no overloading opretaor in java

            {

                arr1[i]=arr[i];//cant save in same array due to immutable nature of strings

            }

        }

        // System.out.println(j);

        for( int i = 0 ;i<j ;i++)

        {

            System.out.println(arr1[i]);

        }

    }

}

Q7

import org.w3c.dom.ls.LSInput;

public class q7 {

    public static void main(String[] args) {

        int n =7;

        for(int i=0;i<7;i++)

        {

            for(int j=6-i;j>0;j--)

            {

                System.out.print(" ");

            }

            for(int k =0;k<=i;k++)

            {

                System.out.print((int)Math.pow(2,k)+" ");

            }

            for(int l =0;l<=i-1;l++)

            {

                System.out.print((int)Math.pow(2,(i-1)-l)+" ");

            }

            System.out.println();

        }

    }

}

Q8

import java.util.Scanner;

class q8{

char answer[][] = new char[8][10];

char key[] = {'D','B','D','C','C','D','A','E','A','D'};

public void getAnswer()

{

Scanner sc = new Scanner(System.in);

for (int i = 0; i < 8; i++) {

System.out.println("Enter answer of Student - "+(i+1));

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

answer[i][j] = sc.next().charAt(0);

}

}

}

void checkAnswer()

{

for (int i = 0; i < 8; i++) {

int count = 0;

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

if(answer[i][j] == key[j])

{

count++;

}

}

System.out.println("Marks of Student - "+(i+1)+" : "+count);

}

}

public static void main(String[] args) {

q8 p = new q8();

p.getAnswer();

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

p.checkAnswer();

}

}

Q9

public class q9 {

    public static void main(String args[]) {

        int a[][] = { {5,3,4,6,7,8,9,1,2},

                {6,7,2,1,9,5,3,4,8}, {1,9,8,3,4,2,5,6,7},

                {8,5,9,7,6,1,4,2,3}, {4,2,6,8,5,3,7,9,1},

                {7,1,3,9,2,4,8,5,6}, {9,6,1,5,3,7,2,8,4},

                {2,8,7,4,1,9,6,3,5}, {3,4,5,2,8,6,1,7,9} };

        System.out.println("Solution is "+ check(a));

    }

    public static boolean check(int arr[][]) {

        int i, j;

        int count[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

        int count1[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

        boolean b = true;

        for (i = 0; i < 9; i++) {

            for (j = 0; j < 9; j++) {

                if (count[arr[j][i]] > i) {

                    b = false;

                    return b;

                }

                if (count1[arr[i][j]] > i) {

                    b = false;

                    return b;

                }

                count1[arr[i][j]]++;

                count[arr[j][i]]++;

            }

        }

        return b;

    }

}

Q10

import java.util.\*;

public class q10

{

    public static StringBuffer encrypt(String text, int s)

    {

        StringBuffer result= new StringBuffer();

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

        {

            if (Character.isUpperCase(text.charAt(i)))

            {

                char ch = (char)(((int)text.charAt(i) +

                                s - 65) % 26 + 65);

                result.append(ch);

            }

            else

            {

                char ch = (char)(((int)text.charAt(i) +

                                s - 97) % 26 + 97);

                result.append(ch);

            }

        }

        return result;

    }

    public static void main(String[] args)

    {

        StringBuffer text= new StringBuffer();

        Scanner sc= new Scanner(System.in);

        System.out.println("Enter text to Encrypt :");

        text.append(sc.next());

        System.out.println("Enter shift :");

        int s = sc.nextInt();

        String str = text.toString();

        System.out.println("Text : " + text);

        System.out.println("Shift : " + s);

        System.out.println("Cipher: " + encrypt(str, s));

    }

}