What is Java?

Java is programming language and a platform. Java is a high level, robust object-oriented and secure programming language.

Platform: Any hardware or software environment in which a program runs, is known as a platform.

Robust: Java is a robust language that can handle run-time errors as it checks the code during the compile and runtime. If any runtime error is identified by the JVM, it will not be passed directly to the underlying system.

Java is popular programming language, created in 1995.

Java was developed by **sun microsystem** in the year 1995**. James gosling** is known as father of Java. Before java, its name was oak.

Why use java?

- Java works on different platforms (windows, mac, Linux etc.)
- It is easy to learn and simple to use
- It is open –source and free
- It is secure, fast and powerful

- It has huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to program and allow code to be reused, lowering development cost.

History of Java:

Java was developed by james gosling who is known as the father of Java, in 1995.

Currently Java is used in internet programming mobile devices games e-business solutions etc.

Java was called initially OAK.

JDK 1.0 was released on Jan 23, 1996.

Java version:

- Jdk 1.0 to Jdk 17
- jdk 18 released by march 22

Feature of Java:

- Simple
- Object oriented
- Portable
- platform independent
- secured
- robust
- High performance

Application:

According the sun, 3 Billion device run Java. There are many devices where Java is currently used.

- Desktop Application
- Mobile Application
- Games
- Embedded System
- Smart Card

Jdk Installation (Java Installations)

Some PCS might have Java Already installed.

To check if you have Java installed on a windows PC, search in the start bar for Java or type the following command Prompt(cmd.exe)

```
C:\Users\> java -version
```

If Java is installed, you will see something like this (depending on version)

```
java version "11.0.1" 2018-10-16 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.1+13-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.1+13-LTS, mixed
mode)
```

If you do not have Java installed on your computer, you can download it for free at **oracle.com.**

Setup for windows:

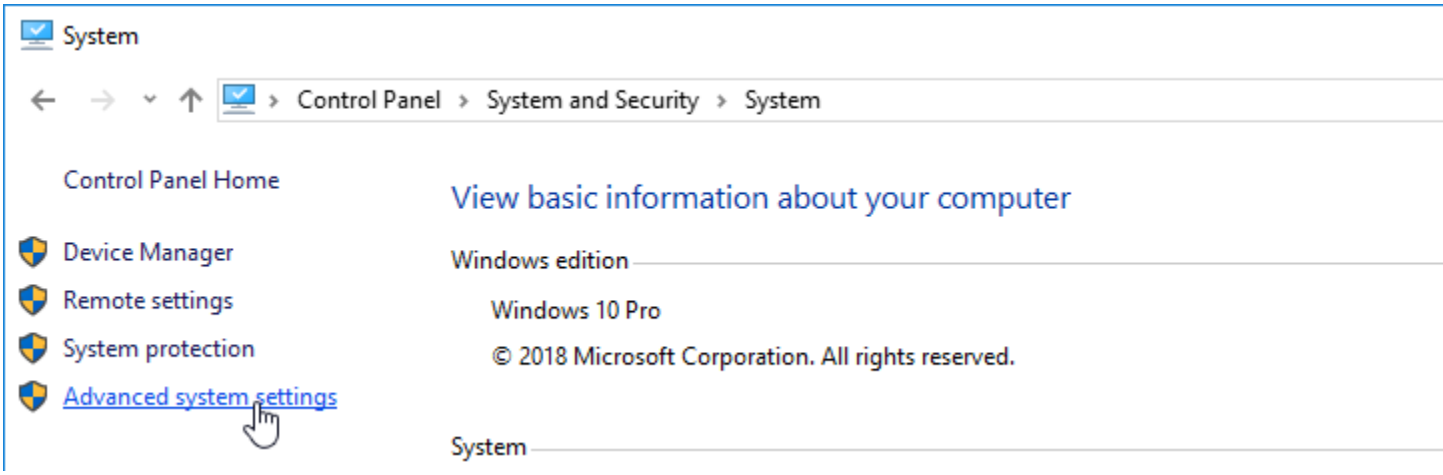
1. Go to "System Properties" (Can be found on Control Panel > System and Security > System > Advanced System Settings).
2. Click on the "Environment variables" button under the "Advanced" tab
3. Then, select the "Path" variable in System variables and click on the "Edit" button
4. Click on the "New" button and add the path where Java is installed, followed by **\bin**. By default, Java is installed in C:\Program Files\Java\jdk-11.0.1 (If nothing else was specified when you installed it).

In that case, You will have to add a new path with: **C:\Program Files\Java\jdk-11.0.1\bin**
Then, click "OK", and save the settings

5. At last, open Command Prompt (cmd.exe) and type **java -version** to see if Java is running on your machine
6. At last, open Command Prompt (cmd.exe) and type **java -version** to see if Java is running on your machine.

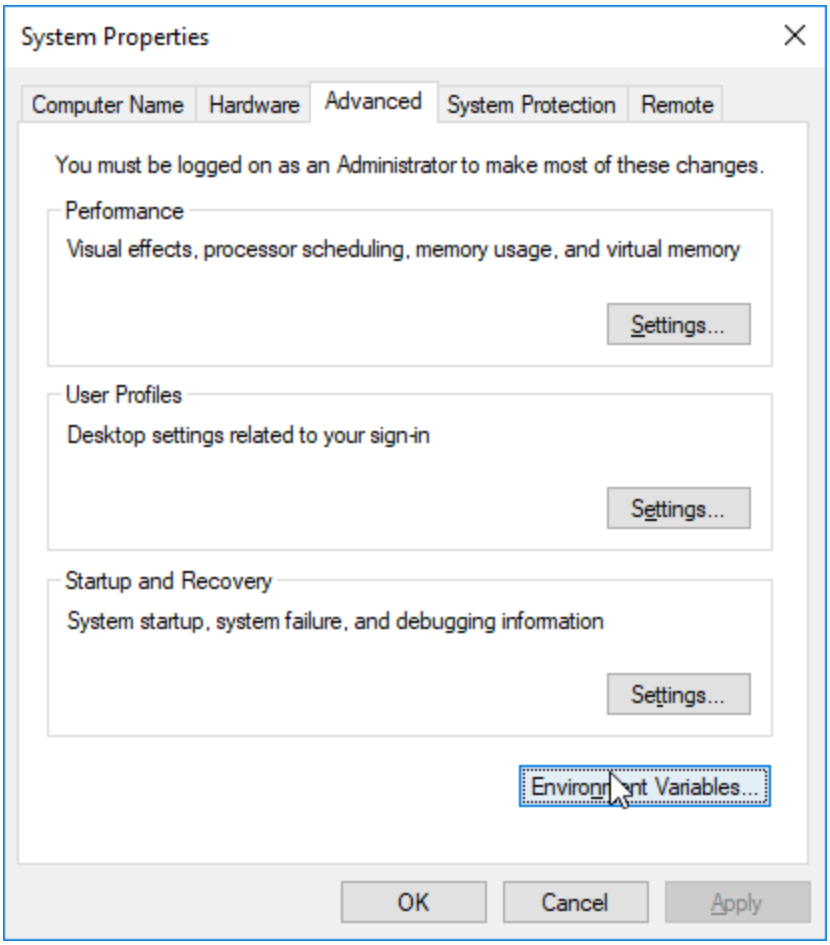
Show how to install Java step-by-step with images »

Step 1



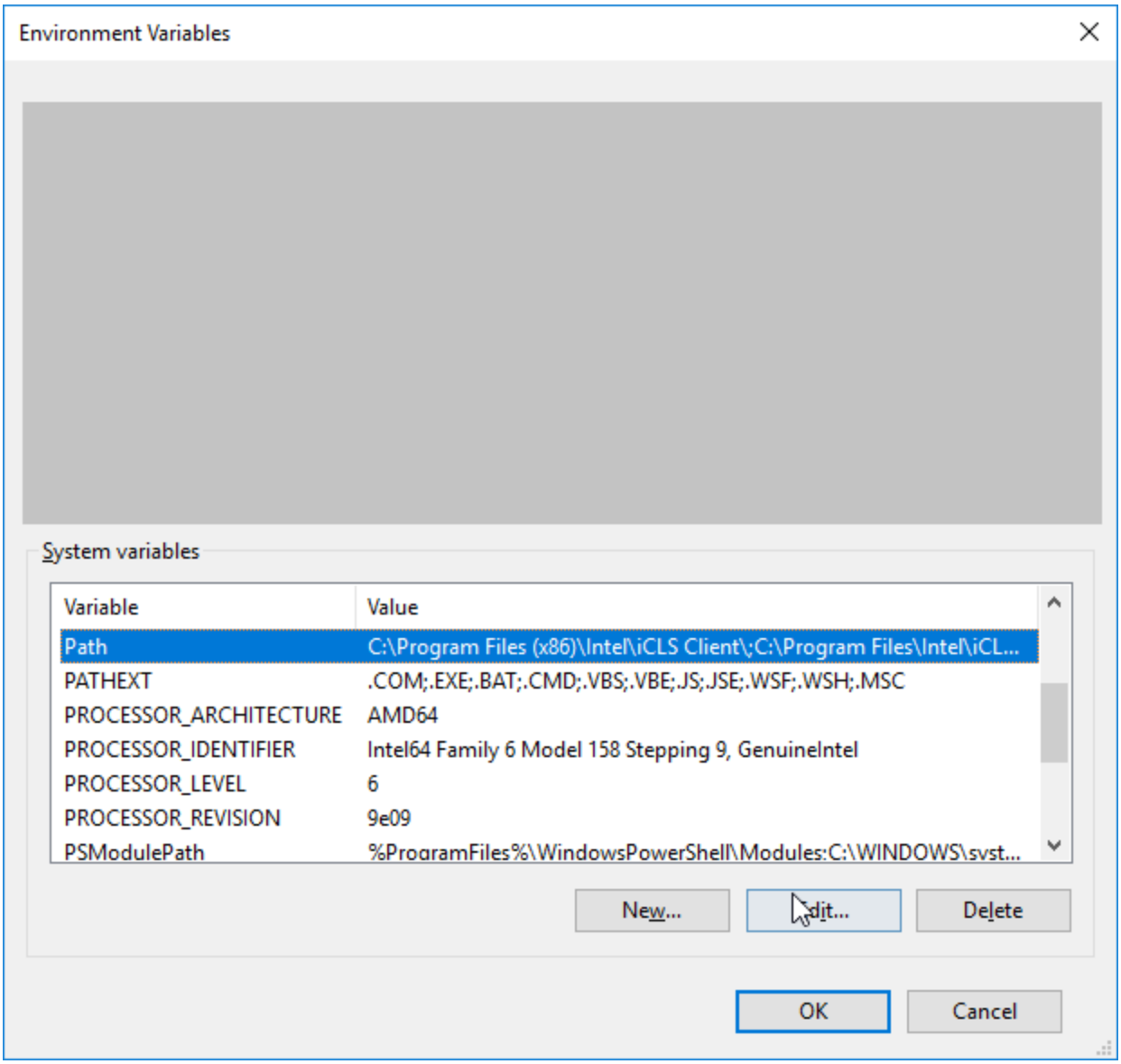
Step 2 »

Step 2


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

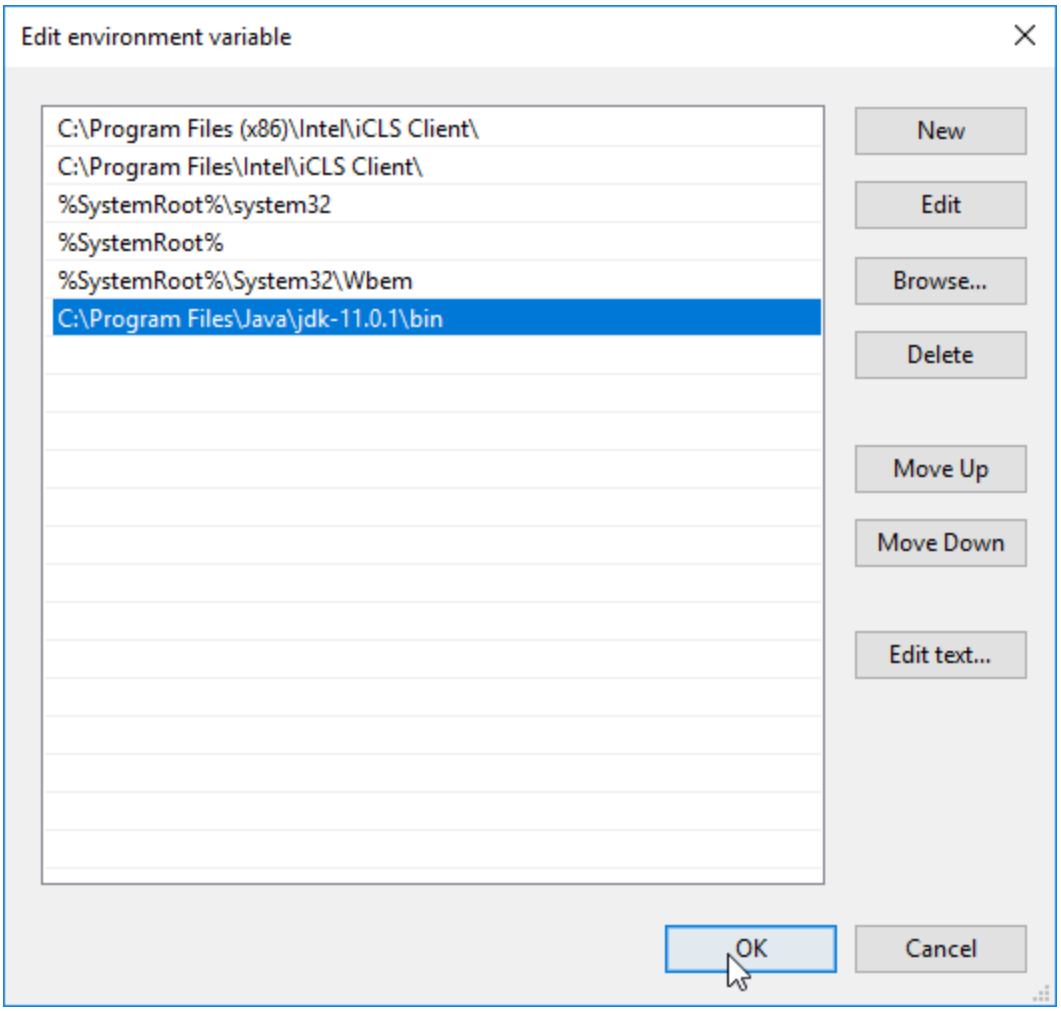
Step 3 »

Step 3


Environment Variables
System variables
Variable
Value
Path
C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCL...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
PROCESSOR_IDENTIFIER
Intel64 Family 6 Model 158 Stepping 9, GenuineIntel
PROCESSOR_LEVEL
6
PROCESSOR_REVISION
9e09
PSModulePath
%ProgramFiles%\WindowsPowerShell\Modules;C:\WINDOWS\syst...
New...
Edit...
Delete
OK
Cancel

Step 4 »

Step 4



Simple Java Program:

Block in Java:

- static block
- non static block
- default constructor
- method
- main method

Global and Local Variable in Java:

Global variable:

global variable has to be declared anywhere in the class body but not inside any method or block. If a variable is declared as global, it can be used anywhere in the class.

Local variable:

If the variable is declared inside a method or block, it is called local variable. Local variable is available only to the method or block in which it is declared.

Package in Java:

Packages in java are used to organize related or similar classes, interface, and enumerations into one group. for example, java.sql package has all classes needed for database operation. Packages are also used to avoid naming conflict between the classes. Using package, you can give same name to different classes.

- packages are declared using keyword ‘package’. They should be declared in the first statement in a java. If you try to declare packages at any other statements, you will get compile time error.

```
package com;

class A{

//some statements

}
```

Import Statement in Java:

If you want to use the member of package in another package, then you have to refer it through package name like below;

```
import pack1.*;
```

Difference between JDK, JVM AND JRE:

JRE:

JRE stands for Java Runtime environment. The Java runtime environment is a set of software tools which are used for developing Java application. It is used to provide the runtime environment. It is implementation of JVM. It physically exits. It contains a set of libraries + other files that JVM uses at runtime.

JDK:

Jdk stands for Java Development kit or sometimes it is also referred as Java Standard Edition Development Kit. JDK is a development environment to develop wide range of application such as desktop application, web application, or mobile application using java programming language.

Jdk contains JRE + development tools

JVM:

JVM(Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exit. It is a specification that provides a runtime environment in which java bytecode can be executed. It can also run those program which are written in other language and compiled to Java byte code

Performs the following task

- Loads code
- verifies code
- execute code
- provides runtime environment

Java Data Types:

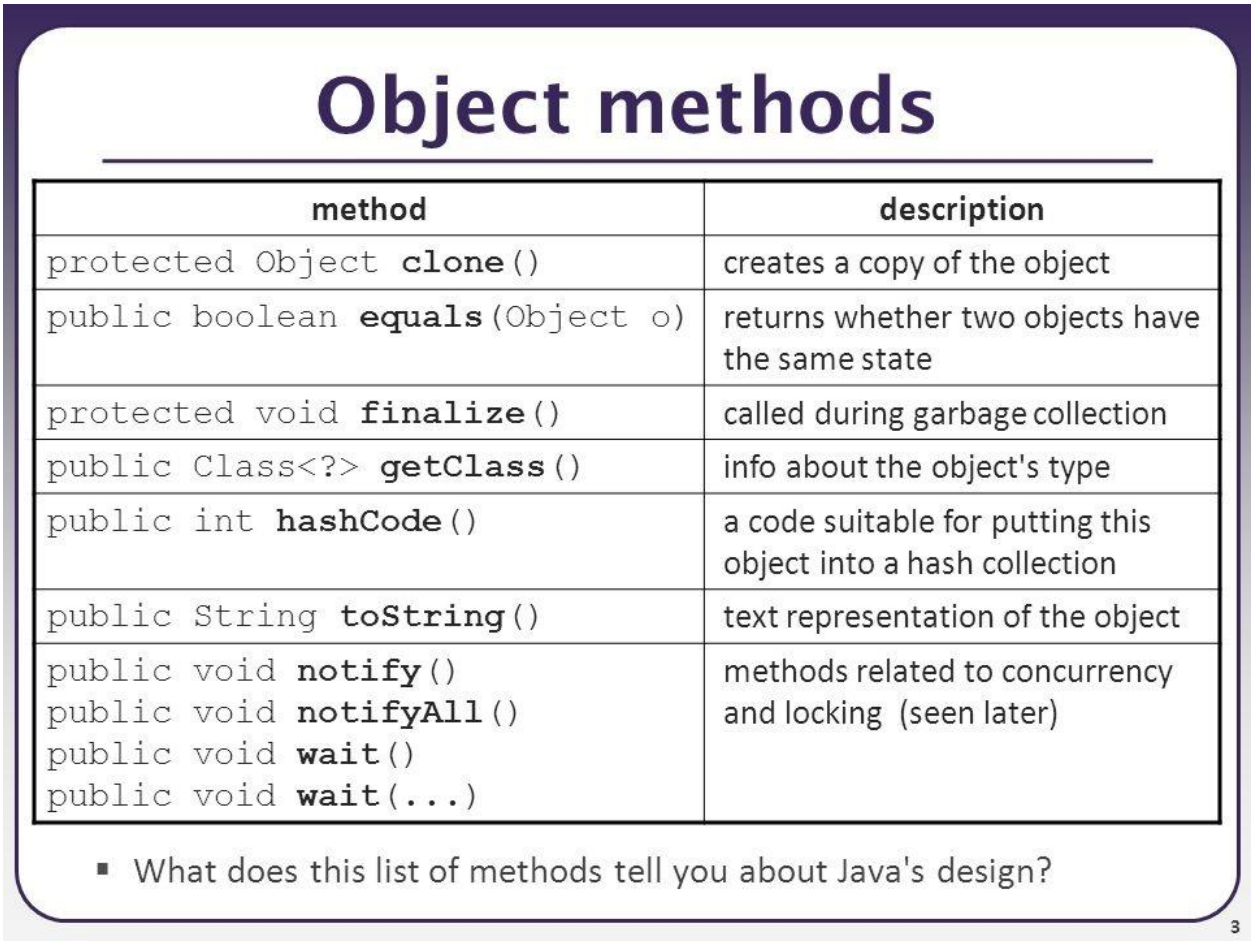
Data types divided into two groups:

1: **Primitive data types**- includes byte, short, int, long, float, double, Boolean and char.

2: Non Primitive data types- String, Array, classes etc.

Object Class in java:

The Object class is the parent class of all the classes in java by default.



Object methods

method	description
`protected Object clone()`	creates a copy of the object
`public boolean equals(Object o)`	returns whether two objects have the same state
`protected void finalize()`	called during garbage collection
`public Class<?> getClass()`	info about the object's type
`public int hashCode()`	a code suitable for putting this object into a hash collection
`public String toString()`	text representation of the object

`public void notify()`
`public void notifyAll()`
`public void wait()`
`public void wait(...)`
	methods related to concurrency and locking (seen later)

- What does this list of methods tell you about Java's design?

3

Setter/Getter:

Getter and setter are used to protect your data, particularly when creating classes.

For each instance variable, a getter method returns its value while a setter method sets or updates its value.

By convention, getter start with the word “get” and setter with the word “set”, followed by a variable name. In both cases the first letter of the variable name is capitalized.

example:

```
public class Vehicle{

private String color ;

//Getter

public String getColor()

{

return color ;

}

//Setter

public void setColor(String c)

{

this.color= c;

}

}
```

---

toString() method in java:

If you want to represent any object as a string, toString() method comes into existence.

The toString() method returns the String representations of the object.

It you print any object; Java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc.

syntax of toString method .

public String toString(){

return roll+" "+name+" "+city;

}

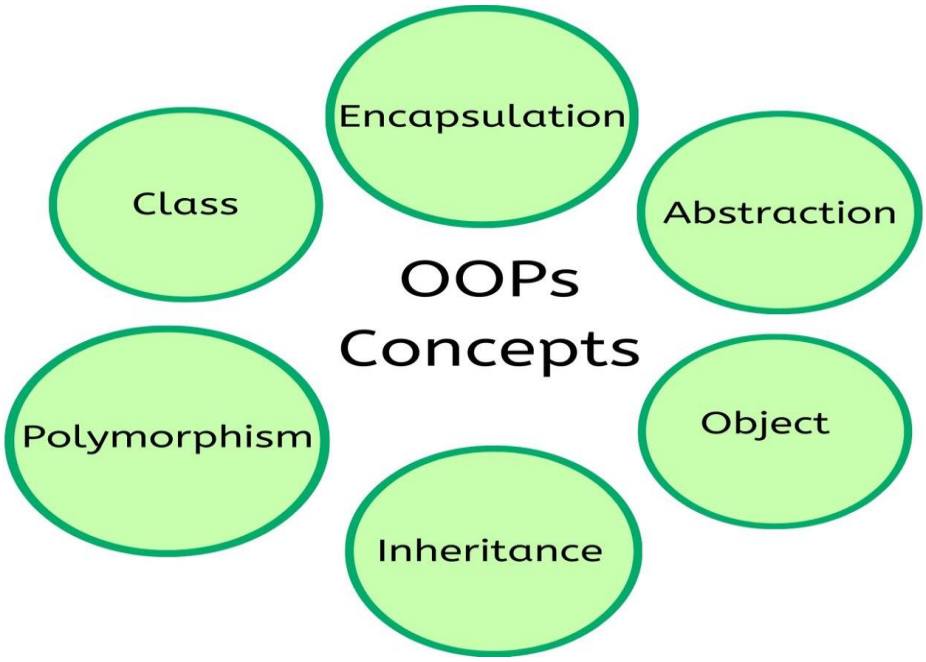
Oops concepts: (Object-Oriented Programming System):

The programming paradigm where everything is represented as an object is known as oop language.

Object means is real-world entity.

example: computer, table.

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation



Object:

An object can be defined as an instance of java class.

- multiple definitions
- An object is a real world entity
- -An object is runtime entity
- The object is runtime entity which has state and behavior
- The object is instance of class

A a = new A ();

Any entity that has state and behavior is known as object. for example, bike, pen.

state- represent the data(value) of an object

Behavior –represent the functionality of an object

Identity – a unique id. it is used internally for jvm to identify the object.

Class:

Collection of object is called class. It is called logical entity.

A class is a template or blueprint from which object are created.

A class contains

- Fields
- Methods
- Constructors
- Blocks
- Nested Class and Interface.

1 :

Inheritance in java:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent objects. It is important part of oops.

Inheritance represents the IS-A relationship.

use of inheritance in java

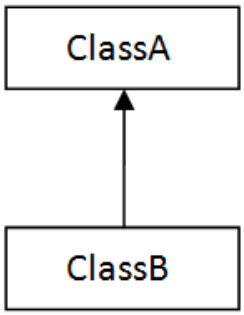
-for code reusability

- for method overriding.

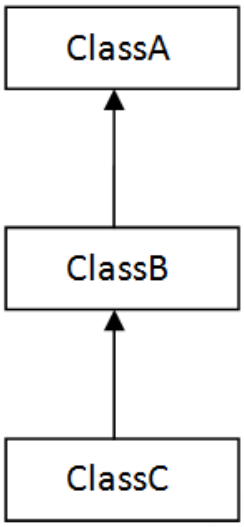
using extends keyword we can achieve inheritance in java.

Types of inheritance in Java:

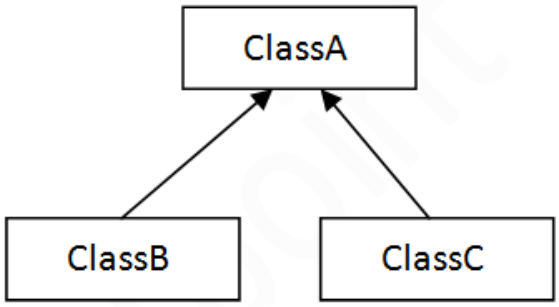
- Single inheritance
- multilevel inheritance
- hierarchical inheritance



1) Single



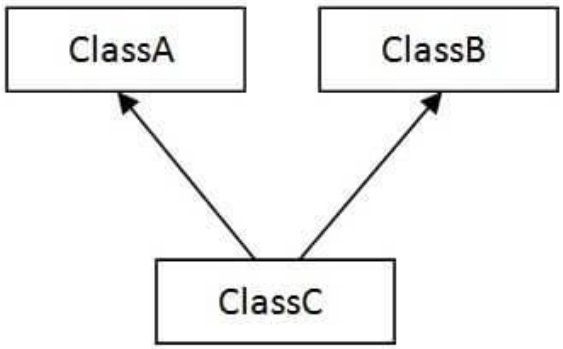
2) Multilevel



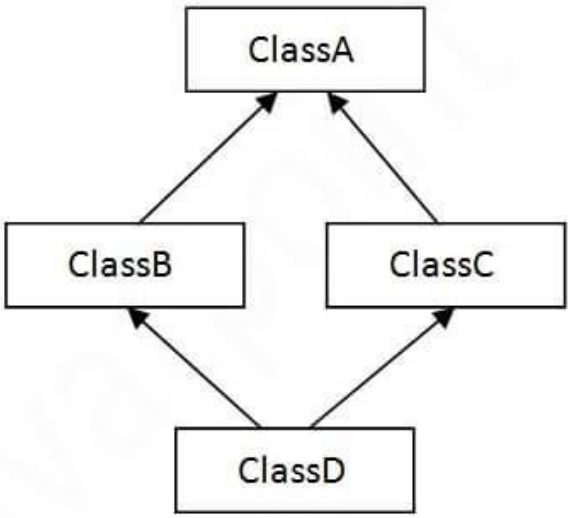
3) Hierarchical

Multiple Inheritance is not supported in Java.

One class inherits multiple classes it is known as multiple inheritance.



4) Multiple



5) Hybrid

.

Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

- **Widening Casting** (automatically) - converting a smaller type to a larger type size
  `byte -> short -> char -> int -> long -> float -> double`
- **Narrowing Casting** (manually) - converting a larger type to a smaller size type
  `double -> float -> long -> int -> char -> short -> byte`

Widening Casting

Widening casting is done automatically when passing a smaller size type to a larger size type:

```
public class Main {

  public static void main(String[] args) {

    int myInt = 9;

    double myDouble = myInt; // Automatic casting: int to double


    System.out.println(myInt);      // Outputs 9

    System.out.println(myDouble);   // Outputs 9.0

  }

}
```

Narrowing Casting

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

```
public class Main {
  public static void main(String[] args) {
    double myDouble = 9.78d;
    int myInt = (int) myDouble; // Manual casting: double to int

    System.out.println(myDouble);   // Outputs 9.78
    System.out.println(myInt);      // Outputs 9
  }
}
```

Polymorphism in Java:

In greek, ploy means many and morph means shapes or forms. So polymorphism refers to any entity which taken many form.

If one task is performed in different different ways, it is called polymorphism.

Example:

There are two types of polymorphism in Java.

- Static polymorphism / method overloading / static binding / early binding / compile time polymorphism.

- Dynamic polymorphism / method overriding / dynamic binding /late binding /run time polymorphism.

Method Overloading:

We can have different forms of same method in the same class is called method overloading.

Requirement for the method overloading:

- method name same and parameter is always different.

Rule for method overloading:

- method overloading always happen in the same class.
- access modifiers doesn't matter in the method overloading.
- return types doesn't matter in the method overloading.
- we can overload main, static, final and private method in java.

Method Overriding:

When a super class method is modified in the sub class, then we call this as method overriding.

Requirement for the method overriding:

- Need parent child relationship.

Rules for method overriding:

- Need parent child relationship.
- the method parameter is always same.
- must have same return type or covariant return is also work.
- access modifiers same or greater than.
- we can't override private, final, and static method.

Access modifiers in Java:

There are four types of Java access modifiers.

-Private: The private access modifier is accessible only within the class.

-Default: If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.

-Protected: The protected access modifier is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class

-Public: The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.

Final keyword in Java:

basically it is use for to restrict the user data.

we can use final keyword before

-class

-data member (variable)

-method

Static keyword in Java:

The static keyword is use for memory management.

We can use static keyword before

-variable (Static variable)

-method

-Block

-Nested Class

Super and this Keyword in java:

Super: Super is used to refer immediate parent class.

uses:

- refer immediate parent class instance variable
- invoke the parent class method
- invoke the parent class constructor

This Keyword in java:

-current object

use:

instance variable

current class method

current class constructor

Abstraction in Java:

Hiding internal details and showing functionality is known is abstraction.

For example: sending text message.

Real time definitions: Showing essential data at essential time is known as abstraction.

for example: student example (i.e. 10th mark,12th mark, height, weight)

- **By using abstract class and Interface we can achieve abstraction in java.**

Abstract class in Java:

A class which is declared as abstract is called as **abstract** class. It can have abstract and non-abstract methods.

- By using abstract class, we can achieve partial abstraction
- It can have abstract and non-abstract method
- It cannot be instantiated

Interface in Java:

An interface is Java is blueprint of a class.

- Interface is used to defined user defined data type
- By using interface, we can achieve 100% abstraction.
- By using interface, we can multiple inheritance in java
- Interface is also representing the IS-A relationship
- Since java 8 we can have default and static method in the interface
- Since java 9 we can have private method in an interface

Uses of Interfaces:

-to achieve 100% abstraction

-support multiple inheritance

-achieve loose coupling

Difference between abstract class and interface

Abstract class	Interface
default const present	no default const
public protected private	public static final
0-100 % abstraction	100% abstraction
abstract and implemented method	only abstract method 1.7jdk

Marker Interface: It an empty interface. (tag interface)

In simple word empty interface in java is called as marker interface. Example of marker interface is serializable, cloneable and Remote Interface.

Uses of java marker interface:

- used to indicate something to compiler or JVM.So if JVM sees a class is serializable, is has done special operation on it, a similar way, if JVM see one class is implement cloneable. it performs some operation to support cloning.

Encapsulation in Java:

Encapsulation is defined as the wrapping up of data under a single unit is called encapsulation.

It is mechanism that binds together code and the data it manipulates.

Another way to think about the encapsulation is, it a protective shield that prevents the data from being accessed by the code outside this shield.

- Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member from any other class and can be accessed only through any member function of its own class in which it is declared.

Advantages of Encapsulation:

- Data hiding: the user will have no idea about the inner implementation of the class. It will not be visible to the user how the class is storing values in the variable.
- Increased Flexibility
- Reusability
- Testing code easy

Java Serialization and Deserialization:

Serialization in java is an important concept that deals with the conversion of objects into a byte stream to transport the java object from one JVM to the other is called as serialization.

And recreate them to the original from it will called as deserialization.

- Serialization in java is the process of converting the java code object into a byte stream to transfer the object.

Why do we need serialization in java?

- Communication:

Serialization involves the procedure of object *serialization* and *transmission.* This enables multiple computer systems to design, share and execute objects simultaneously.

-Caching:

The time consumed in building an object is more compared to the time required for de-serializing it. Serialization minimizes time consumption by *caching* the giant objects

-Deep Copy:

Cloning process is made simple by using Serialization. An exact *replica* of an object is obtained by serializing the object to a *byte array*, and then de-serializing it.

Cross JVM Synchronization:

The major advantage of Serialization is that it works across different JVMs that might be running on different *architectures* or *Operating Systems.*

Persistenc*e:*

The State of any object can be directly stored by applying Serialization on to it and stored in a *database* so that it can be *retrieved later.*

Aggregation/Composition in Java (HAS-A reln):

if a class have an entity reference it is called as aggregation in java.

Aggregation represent HAS-A relationship in java.

In Java, aggregation represents **HAS-A relationship**, which means when a class contains reference of another class known to have aggregation.

Aggregation is a term which is used to refer **one-way relationship** between two objects. For example, **Student** class can have **reference** of **Address** class but vice versa does not make sense.

In Java, aggregation represents **HAS-A relationship**, which means when a class contains reference of another class known to have aggregation.

The HAS-A relationship is based on usage, rather than inheritance. In other words, class A has-a relationship with class B, if class A has a reference to an instance of class B.

When we aggregations:

- Code reuse is also best achieved by aggregations, when there is no is-a relationship.

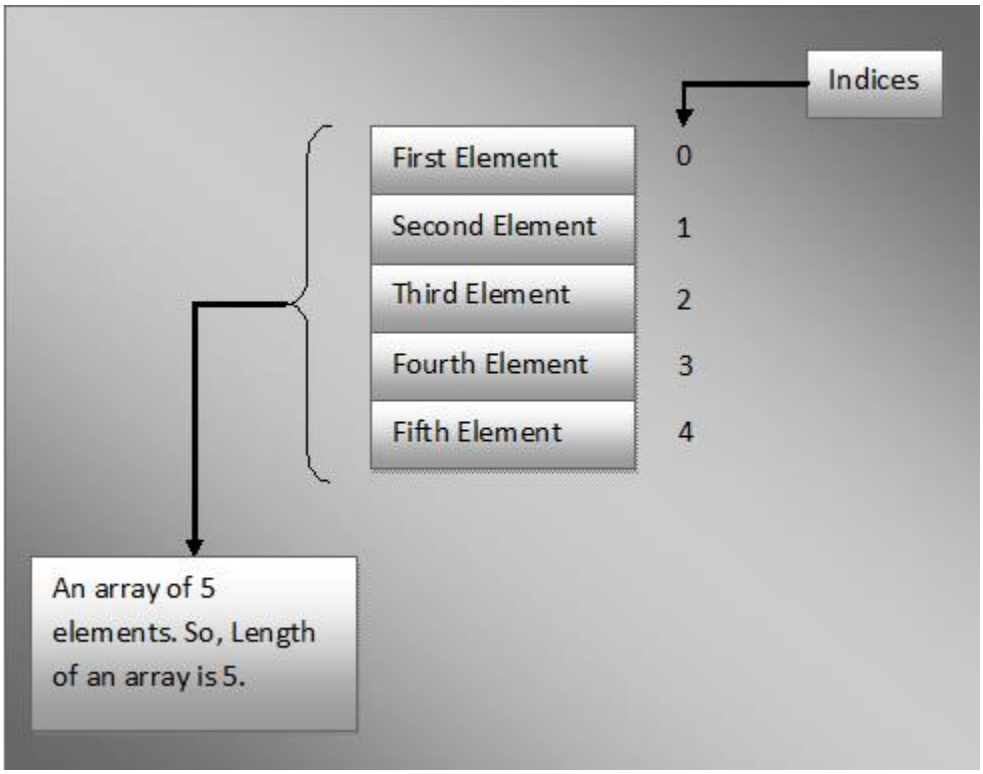
Array in Java:

Array is a set of values where each value is identified by an index.

You can make an array of int's, double's, Boolean's or any other types but all the values of array must be of same type.

The index of an array starts from 0.

The following diagram shows how the array elements are stored in an array object.



Declaring Arrays in Java:

Data_Type[] Variable_Name;

AND

Data_Type Variable_Name[];

```
public class ArraysInJava
{
    public static void main(String[] args)
    {
        int[] arrayOfInts;    //Declaring an array of ints

        double arrayOfDoubles[];   //Declaring an array of doubles

        char[] arrayOfChars;     //Declaring an array of characters

        boolean arrayOfBooleans[];   //Declaring an array of booleans
    }
}
```

Note : As both styles of declaring arrays in java are valid but the style **Data_Type[] Variable_Name** is preferred. The style **Data_Type Variable_Name[]** comes from C/C++ and it is included in java to accommodate C/C++ programmers.

Instantiating an Array Object:

```
int[] arrayOfInts = new int[10];

arrayOfInts[2]=12;

arrayOfInts[5]=56;
```

2nd way :

```
Double[] arrayofDouble = new double[]{12.56,45.84,14.85};

int[] arrayOfints = {12,21,0,5,7};//this is also ok
```

Accessing Array Elements:

int[] arrayOfints = {12,21,0,5,7};

Sop(arrayOfints[0]);

Sop(arrayOfints[3]);

String Class in Java:

String is a sequence of characters. In java, objects of string is **immutable** which means a constant and cannot be changed once created.

Creating a String

There are two ways to create string in java

- String literal

  String s =”abc”;
- Using new Keyword

  String s = new String(“Java”);

Java String class methods:

- length() it return string length
- substring()
-

Java String compare:

we can compare string in Java on the basis of content and reference.

There are three ways to compare string in java:

- By using equals() method
- By using == operator
- By compareTo() method

Exception handling in Java:

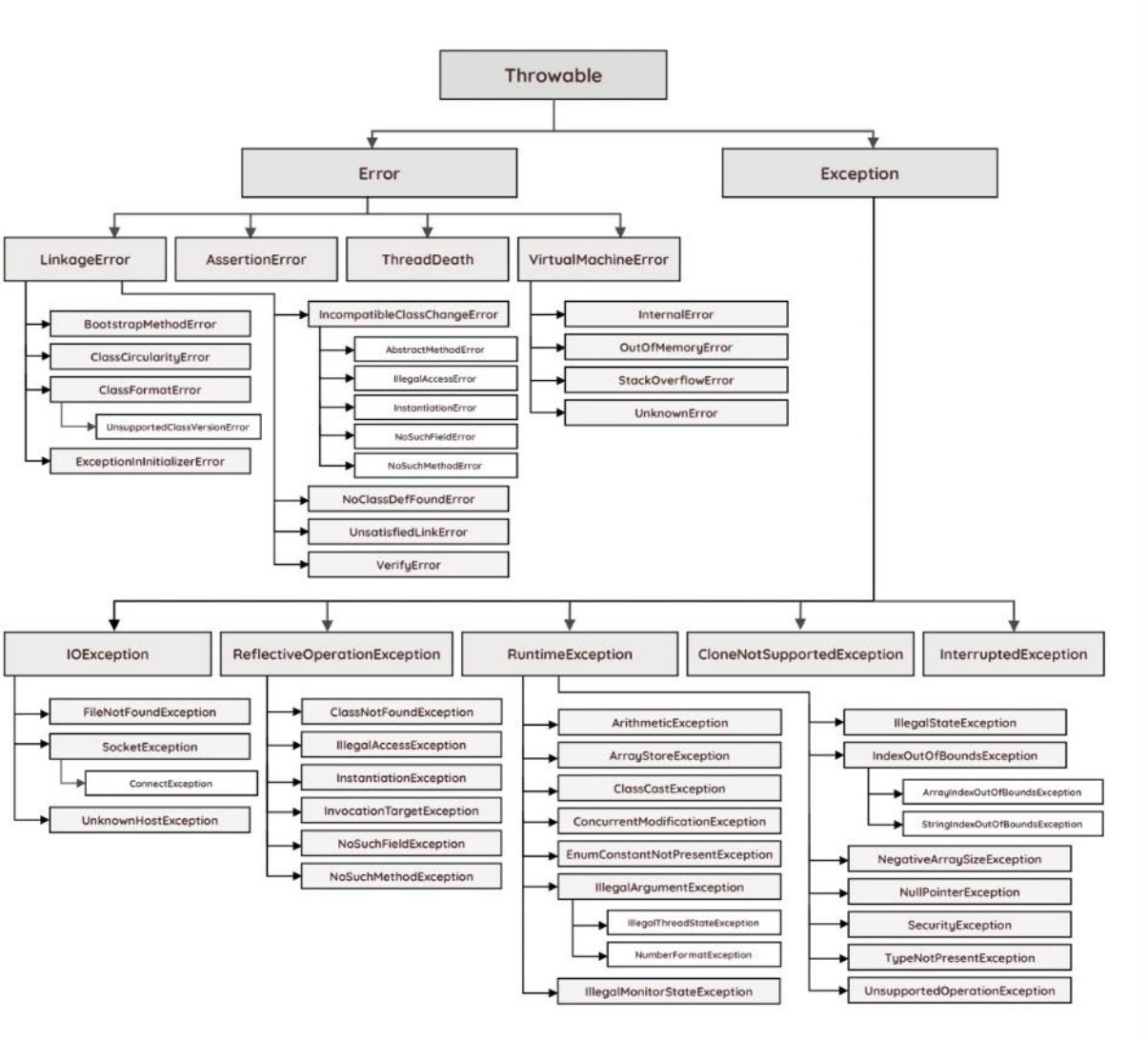
What is an Exception

- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program instructions.
- An exception is an event which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

What is Exception handling?

Exception handling is the mechanism to handle runtime errors such as classnotfoundexception, IOException, SQLException etc.

Hierarchy of Java Exception Classes:



Types of exception:

- Checked Exception (Compile time exception)
- Unchecked Exception (Run time exception)

Checked Exception:

the classes that directly inherit the throwable class except Runtime Exception and error are known as checked exception. Checked exceptions are checked at compile-time.

Unchecked Exception:

The classes that inherit the runtime exception are known as unchecked exception. Unchecked exception is not checked at compile time they are checked at run time.

Ways to handle Exception in java:

- Try
- Catch
- finally
- throw
- throws

Try Catch Block:

try{

This is the try block

In this block , keep those statement which may throw run time exception

}

Catch{

This is the catch block ,

It takes one argument of type java.lang.Exception

This block catches the exception thrown by try block

}

finally {

This is finally block

}

Nested Try catch block in Java:

In java try catch blocks can be nested i.e one try block can contain another try-catch block.

```
try{ //outer try block
//some statement here

Try{ //inner try block

}catch(){//inner catch block

}

}catch(){//outer catch block

}
```

Multiple catch block in java:

In some cases, a single statement may throw more than one type of exception. In such cases, Java allows you to put more than one catch block. One catch block handles one type of exception. When an exception is thorwn by the try block, all the catch blocks are examined in the order they appear and once catch block which matches with exception thrown will be executed.

Printing Exception message ways:

- printStackTrace() method : It prints the name of the exception, description and complete stack trace including the line where exception occurred .
- getMessage() method: Mostly used, It prints the description of the exception.
- e.toString(): It prints name and description of the exception

Try catch using pipe (|) operator:

From java 7 onwards, there is one way for handling multiple exceptions. Multiple exception thrown by the try block can be handled by a single catch block using pipe (|) operator.

Throwing and Re-Throwing Exception in Java:

- throw keywords
- throws keywords

Throw:

Java throw keyword is used to throw an exception explicitly.

- We can throw either checked or unchecked exception
- It's mainly used to throw a custom exception

  for example

  throw new exception class ("error message");

Throws:

The java throws keyword is used to declare an exception. It gives an information to the programmer that there may occur an exception. So it is better for the programmer to provide the exception handling code so that the normal flow of the program can be maintained.

Java Custom Exception:(User Defined Exception)

In java, we can have defined our own exception classes as per our requirements. These exceptions are called user defined exception in java or customized exceptions.

Collection Frameworks in Java:

Collection Framework in java is a centralized and unified theme to store and manipulate the group of objects. Java Collection Framework provides some pre-defined classes and interfaces to handle the group of objects. Using collection framework, you can store the objects as a **list** or as a **set** or as a **queue** or as a **map** and perform operations like adding an object or removing an object or sorting the objects without much hard work.

Why Collection Framework:

Collections are nothing but group of objects stored in well-defined manner. Earlier, Arrays are used to represent these group of objects. But, arrays are not re-sizable. size of the arrays is fixed. Size of the arrays cannot be changed once they are defined. This causes lots of problem while handling group of objects. To overcome this drawback of arrays, **Collection framework** or simply collections are introduced in java from JDK 1.2.

Class Hierarchy of Collection Framework:

All classes and interfaces related to collection framework are placed **in java.util** package .

java.util.collection interface is at the top class hierarchy of collection framework.

Below diagram shows the class hierarchy of collection framework.

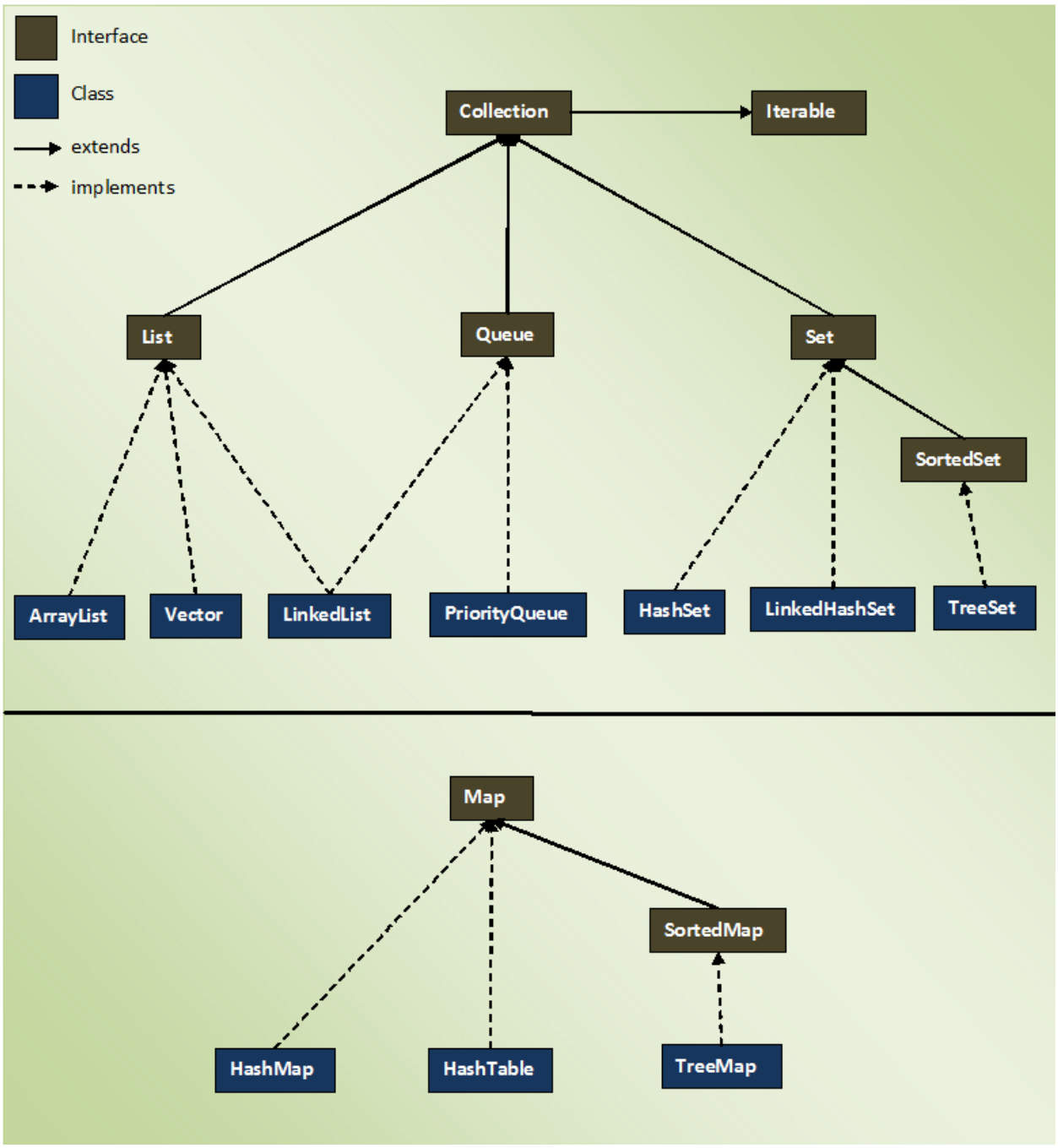
The entire collection is divided into the four interfaces.

1: List

2: Queue

3: Set

4: Map



Collection framework list interface:

List interface represent an ordered or sequential collection of objects.

This interface has some methods which can be used to store and manipulate the ordered collection of objects. The classes which implemented the list interfaces are called as Lists.Arraylist, linkedlist , vector are some examples of list interface .

Properties of List interface:

-elements of the list are ordered using zero based index

- you can access the elements of lists using an integer index

- elements can be inserted at a specific position using integer index. Any pre-existing elements at or beyond that position are shifted right.

- A list may contain duplicate elements

- A list may contain multiple null elements

Internal working of arraylist

```
int newCapacity = (oldCapacity * 3)/2 + 1;
```

The load factor is the measure that decides when to increase the capacity of the ArrayList. The default load factor of an ArrayList is 0.75f. For example, current capacity is 10. So, loadfactor = 10*0.75=7 while adding the 7th element array size will increase. So, It would be good practice if we choose the initial capacity, by keeping the number of expected elements in mind as approx.

Collection framework Set interface:

The set interface defines a set. The set is a liner collection of objects with no duplicates. Duplicate elements are not allowed in a set. The set interface extends collection interface. Set interface does not have its own methods. All its methods are inherited from collection interface. The only change that has been made to set interface is add () method will return false if you try to insert an element which is already present in the set. The classes which implemented the set interfaces are called as hashset, Treeset ,Linkedhashset are some examples of Set interface.

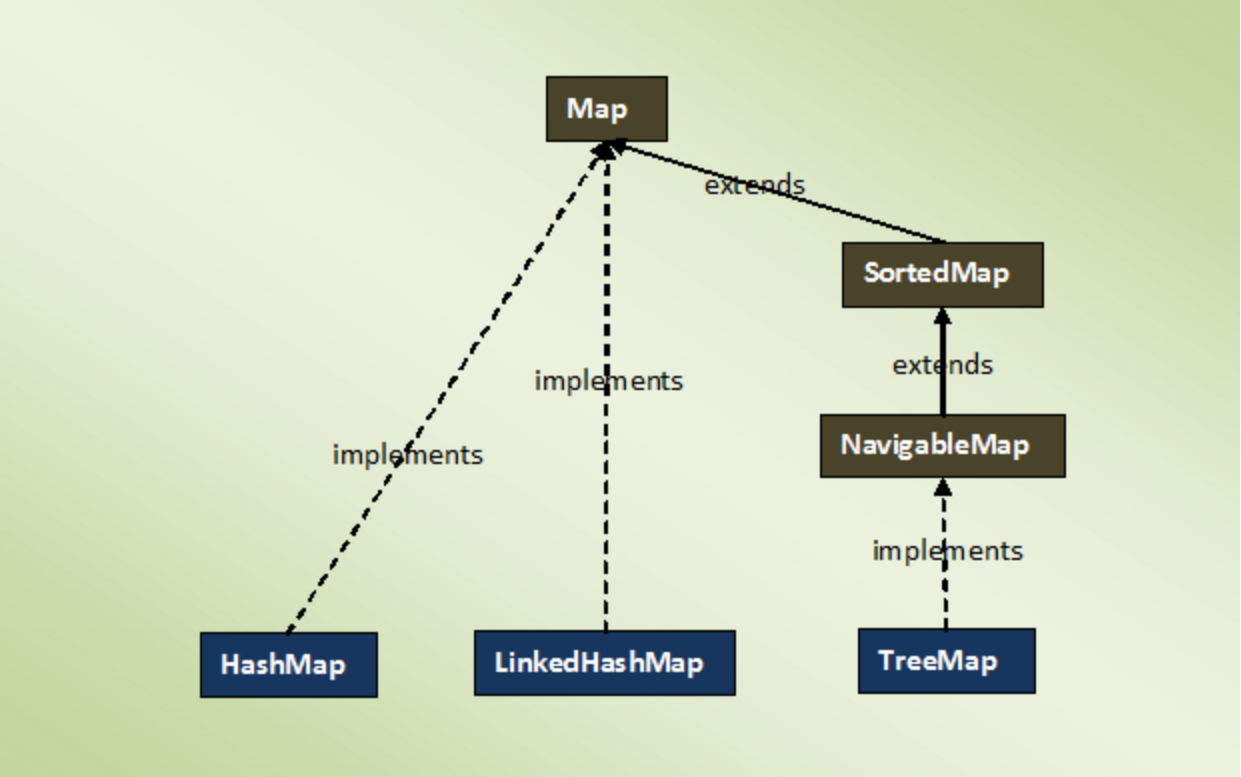
Properties of Set:

- set contain only unique elements. It does not allow duplicate.
- Set can contain only one null element.
- Random access of elements is not possible.

Collection framework Map interface:

- The Map interface in java is one of the four top level interface of Java Collection framework along with List, Set, and Queue interface. But unlike others it doesn't inherit from collection interface. Instead it starts its own interface hierarchy for maintain the key-value associations. Map is an object of key-value associations. Map is an object of key-value pairs where each key is associated with a value. This interface is the replacement of Dictionary class which is an abstract class introduced in JDK 1.0.

Hashmap, LinkedHashMap, and Treemap are three popular implementation of Map interface.
Below picture shows the hierarchy of Map interface in java.



Properties of Map:

- map interface is a part of Java Collection framework, but it doesn't inherit collection Interface.
- Map interface stores the data as a key-value pair where each key is associated with a value.
- A map can't have duplicate keys but can have duplicate values
- each key at must be associated with one value
- each key-value pair of the map are stored as map. Entry objects. Map entry is an inner interface of map interface.
- The common implementations of map interface are Hashmap.LinkedHashmap and Treemap

- Order of elements in map is implementations dependent. Hashmap doesn't maintain any order of elements. Linkedhashmap maintains insertion order of elements. Whereas TreeMap place the elements according to supplied comparator.

Comparable and Comparator Interface:

Java provides two interfaces to sort objects using data members of the class

- Comparable
- Comparator

Comparable:

A comparable object is capable of comparing itself with another object. The class itself must implements the java. lang.Comparable interface to compare its instance.

Comparator:

A comparator interface is used to order the object of user-defined classes. A comparator object is capable of comparing two objects of the same class.

Difference between Comparable and comparator Interface:

KEY DIFFERENCES:

- Comparable in Java is an object to compare itself with another object, whereas Comparator is an object for comparing different objects of different classes.

- Comparable provides compareTo() method to sort elements in Java whereas Comparator provides compare() method to sort elements in Java.
- Comparable interface is present in java.lang package whereas Comparator interface is present in java.util package.
- Comparable provides single sorting sequence while Comparator provides multiple sorting sequences.
- Comparable affects the original class whereas comparator doesn't affect the original class.

JDBC in Java:

Java JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

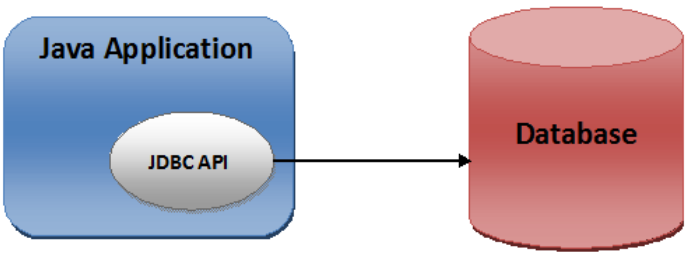
· JDBC-ODBC Bridge Driver,

· Native Driver,

· Network Protocol Driver, and

· Thin Driver

Java application, **JDBC API** and **Database** can be schematically represented as below.



Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

· Connect to the database

· Execute queries and update statements to the database

· Retrieve the result received from the database.

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

· JDBC-ODBC bridge driver

· Native-API driver (partially java driver)

· Network Protocol driver (fully java driver)

· Thin driver (fully java driver)

1) JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

In Java 8, the JDBC-ODBC Bridge has been removed.

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

Advantages

· easy to use.

· can be easily connected to any database.

Disadvantages

· Performance degraded because JDBC method call is converted into the ODBC function calls.

· The ODBC driver needs to be installed on the client machine.

2) Native API driver

The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

Advantages:

· performance upgraded than JDBC-ODBC bridge driver.

Disadvantages

· The Native driver needs to be installed on the each client machine.

· The Vendor client library needs to be installed on client machine.

3) Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

Advantages:

· No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:

· Network support is required on client machine.

· Requires database-specific coding to be done in the middle tier.

· Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

4) Thin driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

Advantages:

· Better performance than all other drivers.

· No software is required at client side or server side.

Disadvantages:

· Drivers depend on the Database.

So we can use Thin driver i.e. JDBC Driver.

The Thin Driver converts JDBC calls directly into the vendor-specific database protocol. That is Why it is known as thin driver. It is fully return in Java Language.

Steps to connect the Database:

There are 5 steps to connect any java application with the database using jdbc. These steps are as follows.

1 Register the Driver class

2 Create the connection

3 Create the statement

4 Execute the queries

5 close the connection

By using above steps we can create the database connectivity.

JDBC API provides 3 different interfaces to execute the different types of SQL queries. They are,

1) ***Statement*** – Used to execute normal SQL queries.

2) ***PreparedStatement*** – Used to execute dynamic or parameterized SQL queries.

3) ***CallableStatement*** – Used to execute the stored procedures.

1) Statement

Statement interface is used to execute normal SQL queries. You can't pass the parameters to SQL query at run time using this interface. This interface is preferred over other two interfaces if you are executing a particular SQL query only once. The performance of this interface is also very less compared to other two interfaces. In most of time, Statement interface is used for DDL statements like **CREATE**, **ALTER**, **DROP** etc.

2) PreparedStatement

PreparedStatement is used to execute dynamic or parameterized SQL queries. PreparedStatement extends Statement interface. You can pass the parameters to SQL query at run time using this interface. It is recommended to use PreparedStatement if you are executing a particular SQL query multiple times. It gives better performance than Statement interface. Because, PreparedStatement are precompiled and the query plan is created only once irrespective of how many times you are executing that query. This will save lots of time.

3) CallableStatement

CallableStatement is used to execute the stored procedures. CallableStatement extends PreparedStatement. Usng CallableStatement, you can pass 3 types of parameters to stored procedures. They are : ***IN*** – used to pass the values to stored procedure, ***OUT*** – used to hold the result returned by the stored procedure and ***IN OUT*** – acts as both IN and OUT parameter. Before calling the stored procedure, you must register OUT parameters using ***registerOutParameter()*** method of CallableStatement. The performance of this interface is higher than the other two interfaces. Because, it calls the stored procedures which are already compiled and stored in the database server.

```
DELIMITER $$


DROP PROCEDURE IF EXISTS `get_merit_student` $$

CREATE PROCEDURE `b21`.`get_merit_student`(IN prd int(10), IN prc varchar(40))

BEGIN

    insert into  student values(prd,prc);


END $$


DELIMITER ;
```

Statement	PreparedStatement	CallableStatement
It is used to execute normal SQL queries.	It is used to execute parameterized or dynamic SQL queries.	It is used to call the stored procedures.
It is preferred when a particular SQL query is to be executed only once.	It is preferred when a particular query is to be executed multiple times.	It is preferred when the stored procedures are to be executed.
You cannot pass the parameters to SQL query using this interface.	You can pass the parameters to SQL query at run time using this interface.	You can pass 3 types of parameters using this interface. They are – IN, OUT and IN OUT.
This interface is mainly used for DDL statements like CREATE, ALTER, DROP etc.	It is used for any kind of SQL queries which are to be executed multiple times.	It is used to execute stored procedures and functions.
The performance of this interface is very low.	The performance of this interface is better than the Statement interface (when used for multiple execution of same query).	The performance of this interface is high.

SQL:

Introduction to SQL

SQL is a standard language for accessing and manipulating databases.

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases

- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Some of The Most Important SQL Commands

- `SELECT` - extracts data from a database
- `UPDATE` - updates data in a database
- `DELETE` - deletes data from a database
- `INSERT INTO` - inserts new data into a database
- `CREATE DATABASE` - creates a new database
- `ALTER DATABASE` - modifies a database
- `CREATE TABLE` - creates a new table
- `ALTER TABLE` - modifies a table
- `DROP TABLE` - deletes a table
- `CREATE INDEX` - creates an index (search key)
- `DROP INDEX` - deletes an index

SQL SELECT Statement

The SQL SELECT Statement

The `SELECT` statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SQL SELECT DISTINCT Statement

The SQL SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT Country FROM Customers;

SELECT COUNT(DISTINCT Country) FROM Customers;

SELECT Count(*) AS DistinctCountries
FROM (SELECT DISTINCT Country FROM Customers);
```

SQL WHERE Clause

SQL PRIMARY KEY Constraint

The `PRIMARY KEY` constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL FOREIGN KEY Constraint

The `FOREIGN KEY` constraint is used to prevent actions that would destroy links between tables.

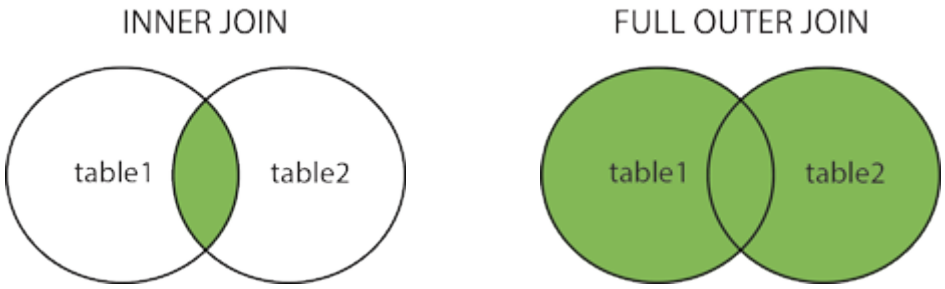
A `FOREIGN KEY` is a field (or collection of fields) in one table, that refers to the `PRIMARY KEY` in another table.

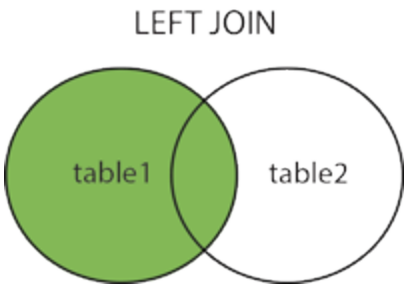
The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

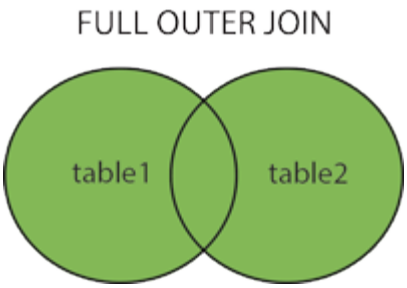
Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table







MQ SQL QUERIES :

SELECT * FROM EMPLOYEE;

USE b14batch;

SELECT * FROM EMPLOYEE;

CREATE TABLE EMPLOYEE(EMPID int , EMPNAME VARCHAR(100) , EMPADDRESS VARCHAR(100), EMPSALARY VARCHAR(200));

INSERT INTO EMPLOYEE VALUES(101,'Rahul','PUNE',20000);

INSERT INTO EMPLOYEE VALUES(105,'VISHAL','PCMC',30000);

INSERT INTO EMPLOYEE VALUES(106,'Test','MUMBAI',70000);

INSERT INTO EMPLOYEE VALUES(107,'Rahul','NASHIK',50000);

```
SELECT EMPID,EMPNAME from EMPLOYEE;

SELECT COUNT(*) tablerecordcount  from EMPLOYEE;

SELECT * FROM EMPLOYEE e  WHERE e.EMPSALARY >=30000;

SELECT * FROM EMPLOYEE e  WHERE e.EMPSALARY <= 30000;

SELECT * FROM EMPLOYEE e WHERE e.EMPSALARY> 30000 and e.EMPNAME like '%R%';

SELECT * FROM EMPLOYEE e WHERE e.EMPSALARY> 30000 OR e.EMPNAME like '%R%';

SELECT DISTINCT EMPNAME FROM EMPLOYEE ;

SELECT COUNT(DISTINCT EMPNAME) tablerecordcount  from EMPLOYEE;

SELECT COUNT(EMPID) , EMPNAME FROM EMPLOYEE GROUP BY EMPNAME ;

SELECT COUNT(EMPNAME ) AS tt, EMPNAME  from EMPLOYEE  GROUP BY EMPNAME  having tt>1  ;

TRUNCATE TABLE EMPLOYEE ;

DROP TABLE EMPLOYEE ;
```

---

use b14batch;

```
USE b14batch;

SELECT * FROM EMPLOYEE;
```

```
CREATE TABLE EMPLOYEE(EMPID int , EMPNAME
VARCHAR(100) , EMPADDRESS VARCHAR(100), EMPSALARY
VARCHAR(200));


INSERT INTO EMPLOYEE VALUES(101,'Rahul','PUNE',20000);

INSERT INTO EMPLOYEE VALUES(105,'VISHAL','PCMC',30000);

INSERT INTO EMPLOYEE VALUES(106,'Test','MUMBAI',70000);

INSERT INTO EMPLOYEE VALUES(107,'Rahul','NASHIK',50000);


INSERT INTO EMPLOYEE(EMPID.ENAME) values(108,'JJ');



UPDATE EMPLOYEE SET EMPNAME='PARTH' WHERE EMPID =
101;
```

```
UPDATE EMPLOYEE SET EMPADDRESS='AHMEDNAGAR' WHERE
EMPID = 107;

UPDATE EMPLOYEE e SET EMPADDRESS='PUNE',EMPNAME=''
WHERE e.EMPSALARY>30000;

UPDATE EMPLOYEE SET EMPID=109,EMPNAME='Digambar' ,
EMPSALARY='55000' WHERE EMPADDRESS='AHMEDNAGAR';

DELETE FROM EMPLOYEE WHERE EMPID=106;

TRUNCATE EMPLOYEE ;

DROP EMPLOYEE;
```

```
CREATE TABLE course (
   course_id    INT NOT NULL PRIMARY KEY,
   course_name  VARCHAR(20) NOT NULL
);



select * from COURSE;

INSERT INTO COURSE VALUES(101,'JAVA');
insert into course values(102,'Cpp');
insert into course values(103,'Python');
insert into course values(107,'TESTING');



CREATE TABLE faculty (
   faculty_id    INT NOT NULL PRIMARY KEY,
```

```
    faculty_name  VARCHAR(20) NOT NULL,
    courseid      INT NOT NULL,
    CONSTRAINT fk_faculty FOREIGN KEY ( courseid )
       REFERENCES course ( course_id )
);



select * from  faculty ;

insert into faculty values(201,'ajay',102);

insert into faculty values(202,'nilesh',101);

insert into faculty values(207,'nilesh',103);



CREATE TABLE batch (
   batch_id    INT NOT NULL PRIMARY KEY,
   batch_name  VARCHAR(20) NOT NULL,
   facultyid      INT NOT NULL,
   CONSTRAINT fk_batch FOREIGN KEY ( facultyid )
```

```
      REFERENCES faculty ( faculty_id )
);



CREATE TABLE std (
   student_id    INT NOT NULL PRIMARY KEY,
   student_name  VARCHAR(20) NOT NULL,
   batchid      INT NOT NULL,
   CONSTRAINT fk_std FOREIGN KEY( batchid )
      REFERENCES batch( batch_id )
);

SELECT * FROM BATCH ;

insert into batch values(301,'morning',201);

insert into batch values(302,'evening',202);

insert into std values(401,'pooja','302');

insert into std values(402,'nagesg','301');
```

```
SELECT * FROM STD;

SELECT * FROM COURSE c INNER JOIN faculty f on c.course_id = f.courseid;

SELECT * FROM COURSE c LEFT JOIN FACULTY f on c.course_id = f.courseid;

SELECT * FROM COURSE c RIGHT JOIN FACULTY f on c.course_id = f.courseid;

select * from course c inner join  faculty f on c.COURSE_ID = f.COURSEID;



select * from course c inner join  faculty f on c.COURSE_ID = f.COURSEID

inner join Batch b on f.FACULTY_ID=b.FACULTYID;



select * from course c inner join  faculty f on c.COURSE_ID = f.COURSEID inner join Batch b on
f.FACULTY_ID=b.FACULTYID inner join std s on b.BATCH_ID=s.BATCHID;
```