Owner of java: Oracle

Who is written in java: James gosling and his team

What is java?

Java is a programming language and a platform Independent.java is a high level programming language. Robust secured and object oriented programming language.

It is used to develop application software.

Example Application software:

Google Drive, YouTube, Whats app, Facebook, skype, twitter.

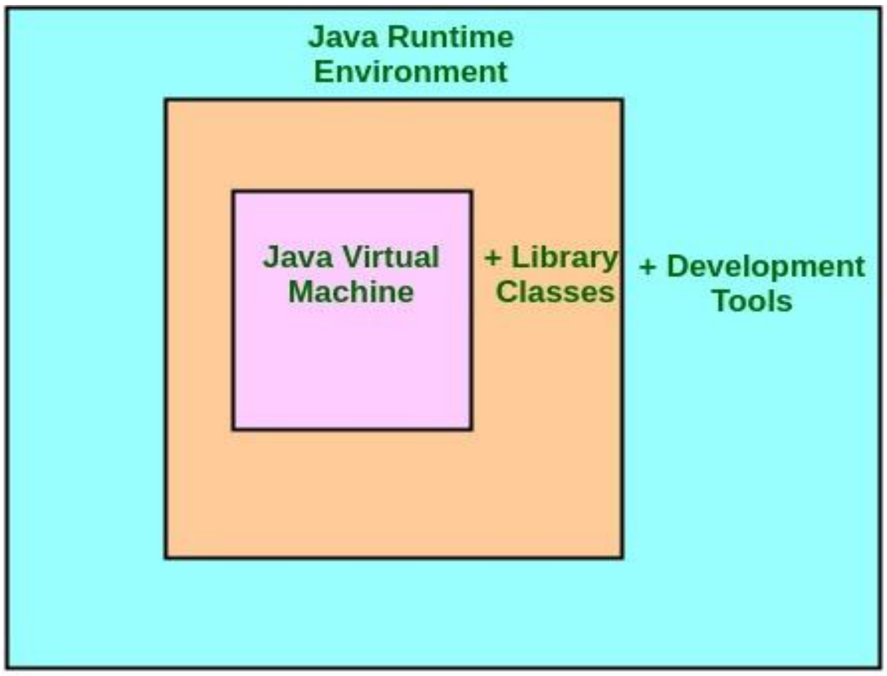
Software Programming language:

C, C++, Java, Php, Dotnet, phython, Html, Forton.

Types of java application:

1).Standalone application ->develop desktop application Example:VLC,ANTIVIRUS,ADOBEREADER	Java Edition J2SE Java 2 standard Edition (Core java)
2).Web application -> develops enterprise application Example: Gmail,Facebook,Flipkart,amazon.com,flipkart.com, Sbionline.com	J2EE Java 2 Enterprise Edition. (Advanced java)
3).Mobile application Example: Mobile apps,games	J2ME Java 2 Micro Edition (Embedded java)

Difference between JDK, JRE and JVM?



JDK = JRE + Development Tool
JRE = JVM + Library Classes

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).**Note :** JDK is only used by Java Developer

JRE - Java Runtime Environment (to say JRE) is an installation package which 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 JDK and 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.

What are the Major features of JDK7?

- Binary Literals.
- Strings in switch Statement.
- Try with Resources or ARM (Automatic Resource Management)
- Multiple Exception Handling.
- Suppressed Exceptions.
- underscore in literals.
- Type Inference for Generic Instance Creation using Diamond Syntax.

What are the Major features of JDK8?

- forEach() method in Iterable interface.
- default and static methods in Interfaces.
- Functional Interfaces and Lambda Expressions.
- Java Stream API for Bulk Data Operations on Collections.
- Java Time API.
- Collection API improvements.
- Concurrency API improvements.
- Java IO improvements.

Why Java is platform independent?

Platform independent practically means "write once run anywhere". Java is platform independent With the help of Jvm , because byte code can be executed at any platform.

Explain what access modifiers can be used for methods?

We can use all access modifiers public, private, protected and default for methods.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

What is the scope of variables in Java in following cases?

1. **Instance Variable:** The member variables must be declared inside class (outside any function). They can be directly accessed anywhere in class by creating an object. Initial values is assigned by default.
2. **Local Variables** (Method Level Scope): Variables declared inside a method have method level scope and can't be accessed outside the method. No default variable for local variable, we need to assign before use.
3. **Static Variables:** A variable which is declared in the class level with static keyword and no need to create object to access that variables. Common properties of all objects are declared as static variable. For example college is common for all student objects.

Code:

```
Public class test {
Static int pincode=609117;    //static Variable
String name= "Manjula";       //Instance Variable

Public void pupAge() {
int age = 10;                  //Local Variable
age = age + 11;
System.out.println("Puppy age is : " + age);
}

Public static void main(String args[]) {
test t = new test();
t.pupAge();
System.out.println(t.name);
System.out.println(pincode);
}
}
```

What is String?

A string is an object that represents a sequence of characters with in doublequots. java. lang. String class is used to create string object.

There are two ways to create a String object:

By string literal: Java String literal is created by using double quotes.
For Example: String s="Welcome";

By new keyword: Java String is created by using a keyword "new".
For example: String s=new String ("Welcome");
It creates two objects (in String pool and in heap) and one reference variable where the variable's will refer to the object in the heap.

Code:

```
Public class test {
Public static void main(String args[]){
String s1="Welcome";//creating string by java string literal
String s2=new String("Welcome");//creating java string by new keyword
System.out.println(s1);
System.out.println(s2);
}
}
```

What is Java String Pool?

String pool is a memory which resides in the heap area of JVM which is specifically used to store objects in Java.

Why String is Immutable?

In java, **string** objects are **immutable**. **Immutable** simply **means** unmodifiable or unchangeable. Once **string** object is created its data or state can't be changed but a new **string** object is created.

```
class Testimmutablestring{
 public static void main(String args[]){
   String s="Manju";
   s.concat(" Manju");   //concat() method appends the string at the end
   System.out.println(s);
    //will print Manju because strings are immutable objects
 }}
```

What is Stack Memory?

- Stack in java is a section of memory which contains **methods, local variables,** and **reference variables.**
- Stack memory is always referenced in **Last-In-First-Out** order.

Explain where variables are created in memory?

- Variables are created in stack. So when the variable is out of scope those variables get garbage collected.

What are the String Methods in java?

1) length()
2) compareTo()
3) concat()
4) IsEmpty()
5) Trim()
6) toLowerCase()
7) toUpper()
8) replace()
9) contains()
10) equals()
11) equalsIgnoreCase()
12) toCharArray()
13) chatAt()
14) indexOf()
15) split()
16) substring()

What is enhanced loop?

Enhanced for loops are used to retrieve all elements from the beginning till ending automatically from an array or collection classes.

Code:

```
class AssignmentOperator {
publicstaticvoid main(String[] args) {
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
for (char item: vowels) {
System.out.println(item);
}
}
}
```

What is foreach in java?
For-each is another array traversing technique

Code:

```
class AssignmentOperator {
publicstaticvoid main(String[] args) {
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
vowels.foreach (c2->system.out.println(c2))
}
}
```

Can we override constructors in java?
Only methods can be overridden in java. Constructors can’t be inherited in java. So there is no constructors Overriding in java.

Can we override static methods in java?

- Static methods can’t be overridden.
- If we have a static method in super class and subclass with same signature then we don’t say that as overriding.
- The concept of over ridding is hiding the base class implementation

What is jar?

- Jar stands for java archive file and package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, **etc**.) into one file for distribution.
- Jars are created by using Jar.exe tool. Jar files contains .class files, other resources used in our application and manifest file.

What is Package?

- Package name should be first line in source code . package name should be declared in small letters. package statement defines the namespace.
- A **java package** is a group of similar types of classes, interfaces and sub-packages.
- Package in java can be categorized in two form, built-in package and user-defined package.

- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Why java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, short which are **not objects.**

What are wrapper classes?

These are known as wrapper classes because they “wrap” the primitive data type into an object of that class.

Code:

```
Public class WrappingUnwrapping {
        Public static void main(String args[])
           {
                int b = 10; // int data type
                //wrapping around Integer object
               Integer intobj = new Integer(b);     // char data type
           }
}
```

What is singleton class and how can we make a class singleton?

- We can make constructor as private. So that We cannot create an object outside of the class. Singleton pattern helps us to keep only one instance of a class at any time.

Code:

```
        package com.myjava.constructors;

        public class MySingleTon {
        private static MySingleTon myObj;
        private MySingleTon(){
        }
        Public static MySingleTon getInstance(){
        if(myObj == null){
        myObj = new MySingleTon();
        }
        returnmyObj;
           }
```

```
Public void getSomeThing(){
System.out.println("I am here....");

Public static void main(String a[]){
MySingleTon st = MySingleTon.getInstance();
st.getSomeThing();
   }
}
```

What are constructors in Java?

- Special method in java.
- Same name as class name
- No return type
- It refers to a block of code which is used to initialize an object
- It invoke when we create an object

There are two types of constructors:

(1) Default constructor – constructor with no arguments

```
class Student3{
int id;
String name;
Public Student3(){
System.out.println("Constructor invoked when we create an object");
}
public static void main(String args[]){
//creating objects
Student3 s1=new Student3();
}
}
```

(2)Parameterized constructor – constructor with arguments

Code:

```
class Student4{
   int id;
   String name;
   //creating a parameterized constructor
   Student4(int i,String n){
   this.id = i;
   this.name = n;
   }
   public static void main(String args[]){
   //creating objects and passing values
```

```
    Student4 s1 = new Student4(111,"Karan");
      }
}
```

Does constructor return any value?

Yes, the constructor implicitly returns the current instance of the class

Is constructor inherited?

No, the constructor is not inherited.

What is super in java?

- The **super** keyword in java is a reference variable that is used to refer parent class objects.
- We cannot use super keyword inside the static block or method.
- Only public and protected methods can be called by the **super keyword**.

Code:

```
class Vehicle
{
    int maxSpeed = 120;
}
  class Car extends Vehicle
{
    int maxSpeed = 180;
    void display()
    {        System.out.println("Maximum Speed: " + super.maxSpeed);
    }
}

  class Test
{
    public static void main(String[] args)
    {
        Car small = new Car();
        small.display();
    }
}
```

What is Lambda Expression?

- It Enable to treat functionality as a method argument, or code as data.
- A function that can be created without belonging to any class.

Syntax:
(argument-list) -> {body}

Java lambda expression is consisted of three components.

1) Argument-list: It can be empty or non-empty as well.
2) Arrow-token: It is used to link arguments-list and body of expression.
3) Body: It contains expressions and statements for lambda expression.

Code:

```
interface Drawable{
public void draw();
}
public class LambdaExpressionExample2 {
public static void main(String[] args) {
int width=10;
//with lambda
Drawable d2=()->{  System.out.println("Drawing "+width); };
d2.draw();
}
}
```

OOPs

List out benefits of object oriented programming language?

- Easy maintenance
- Code reusability
- Code extendibility
- Reliable

What is a class?

- A class is blueprint or template for objects.
- Class consists variables, methods.
- A class tells what type of objects we are creating.

What is an object?

- An Object is instance of class.
- A class defines type of object.
- Every object contains state is called instantiation, initialization and behavior is called method.

Define Inheritance?

- Accessing properties of one class from another class.
- A new class derived from a base class.

What are the types of inheritance?

There are five types of inheritance in Java.

- Single-level inheritance
- Multi-level inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

What is 'IS-A'relationship in java?

(1) 'Is a' relationship is also known as inheritance.

(2) We can implement 'is a' relationship or inheritance in java using extends keyword

Ex: Motor cycle is a vehicle Car is a vehicle Both car and motorcycle extends vehicle.

Code:

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

What is ‘HAS A’’ relationship in java?

- ‘Has a ‘relationship is also known as “composition or Aggregation”.
- Composition – loosely coupled(Car & Indicator)
- Aggregation – tightly coupled(Car & Engine)

(1) Car has a Engine

Code:

```
public class Address {
String city ,state,country;

public Address(String city, String state, String country) {
    this.city = city;
    this.state = state;
    this.country = country;
}
}

public class Emp {
int id;
String name;
Address address;
public Emp(int id, String name,Address address) {
    this.id = id;
    this.name = name;
    this.address=address;
}
void display(){
System.out.println(id+" "+name);
System.out.println(address.city+" "+address.state+" "+address.country);
}
public static void main(String[] args) {
Address address1=new Address("gzb","UP","india");
Address address2=new Address("gno","UP","india");
Emp e=new Emp(111,"varun",address1);
Emp e2=new Emp(112,"arun",address2);
e.display();
e2.display();
}
}
```

What is multiple inheritance? Is it supported by Java?

- If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.
- The problem with multiple inheritance is that if multiple parent classes have a same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.
- Therefore, Java doesn't support multiple inheritance. The problem is commonly referred as Diamond Problem.

Code:

```
interface X
{
  public void myMethod();
}
interface Y
{
  public void myMethod();
}
class JavaExample implements X, Y
{
  public void myMethod()
  {
     System.out.println("Implementing more than one interfaces");
  }
  public static void main(String args[]){
         JavaExample obj = new JavaExample();
         obj.myMethod();
  }
}
```

What is Polymorphism?

One to many form that is, a method has more than one implementation.

What are the types of polymorphism?

There are two types of polymorphism:

1. Compile time polymorphism or Static binding or method overloading
2. Run time polymorphism or Dynamic binding or method overriding

What is method overloading?

1. It must have the same method name.
2. It must have the different arguments.
3. Different Parameters
4. Different types.

Code:

```
Public class test {
Static int add(int a,int b){return a+b;}  //changing no. of arguments
Static int add(int a,int b,int c){return a+b+c;} // changing no. of arguments
Static int add1(int a, int b){return a+b;}  //changing data type of arguments
Static double add1(double a, double b){return a+b;}  //changing data type of
arguments
}

class TestOverloading1{
        public static void main(String[] args){
        System.out.println(test.add1(12,198));
        System.out.println(test.add(190,67,77));
        System.out.println(test.add1(131,114));
        System.out.println(test.add1(1111,1111));
        }
}
```

What is Method Overriding?

In Method Overriding, sub class have the same method with same name and exactly the same number and type of parameters and same return type as a super class.

- Method Overriding is to "Change" existing behavior of method.
- It is a run time polymorphism.
- The implementation of based class method is hidden.

```
class Vehicle{
  //defining a method
  void run(){System.out.println("Vehicle is running");}
}
```

```
//Creating a child class
class Bike2 extends Vehicle{
  //defining the same method as in the parent class
  void run(){System.out.println("Bike is running safely");}
  public static void main(String args[]){
  Bike2 obj = new Bike2();//creating object
  obj.run();//calling method
  }
}
```

Can you override a private or static method in Java?

You cannot override a private or static method in Java. If you create a similar method with same return type and same method arguments in child class then it will hide the super class method.

Can we override java main method?

No, because the main is a static method.

Define Abstraction?

Abstraction is a hiding the method implementation and showing only functionality to the user.

What is the purpose of interface?

1. variable and unimplemented methods.
2. variable considered as public ,static ,final.
3. methods are considered as abstract.

Code:

```
interface MyInterface
{
Public void method1();
Public void method2();
}
class Demo implements MyInterface
{
Public void method1()
  {
        System.out.println("implementation of method1");
  }
Public void method2()
  {
        System.out.println("implementation of method2");
  }
Public static void main(String arg[])
  {
        MyInterface obj = new Demo();
        obj.method1();
  }
}
```

Explain features of interfaces in java?

1) Interfaces cannot be instantiated.
2) We cannot declare static methods inside interface.
3) ‘implements’ keyword is used to implement interface.
4) Unlike class, interface can extend any number of interfaces.
5) We can define a class inside interface and the class acts like inner class to interface.

What are the important features of Abstract class?

The important features of abstract classes are:

1) Abstract classes cannot be instantiated.

2) An abstract classes contains abstract methods, concrete methods or both.

3) Any class which extends abstract class must override all methods of abstract class.

Can we declare a class as Abstract without having any abstract method?

- Yes we can create an abstract class by using abstract keyword before class name even if it doesn't have any abstract method.
- However, if a class has even one abstract method, it must be declared as abstract otherwise it will give an error.

Define Encapsulation?

Encapsulation is wrapping up of data under a single unit .Encapsulation is a data hiding and abstraction.

Code:

```
public class Encapsulate
{
   private String geekName;
   private int geekRoll;
   private int geekAge;

    public void setAge( int newAge)
   {
     geekAge = newAge;
   }

   public int getAge()
   {
     return geekAge;
   }
   public void setName(String newName)
   {
     geekName = newName;
   }
      public String getName()
   {
     return geekName;
   }

}

public class TestEncapsulation
{
   public static void main (String[] args)
   {
      Encapsulate obj = new Encapsulate();

      // setting values of the variables
      obj.setName("Harsh");
      obj.setAge(19);


      // Displaying values of the variables
      System.out.println("Geek's name: " + obj.getName());
      System.out.println("Geek's age: " + obj.getAge());
```

```
        // Direct access of geekRoll is not possible
        // due to encapsulation
        // System.out.println("Geek's roll: " + obj.geekName);
    }
}
```

What is Enumeration?

- Enumeration is a new feature from Java 5.0.
- Enumeration is set of named constants .
- We use enum keyword to declare enumeration.
- The values defined in enumeration are enum constants.
- Each enum constant declared inside a enum class is by default public, static and final.

Example : package java examples; public enum Days { SUN,MON,TUE,WED,THU,FRI,SAT; } SUN,MON,TUE,WED,THU,FRI,SAT are enum constants.

Code:

```
class EnumExample5{
enum Day{ SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDA
Y, FRIDAY, SATURDAY};
public static void main(String args[]){
Day day=Day.MONDAY;
switch(day){
case SUNDAY:
 System.out.println("sunday");
 break;
case MONDAY:
 System.out.println("monday");
 break;
default:
System.out.println("other day");
}
}}
```

Thread

- Subset of a process
- A thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution.
- Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

- New
- Runnable
- Running
- Non-Runnable (Blocked)
- Terminated

Explain the life cycle of thread?

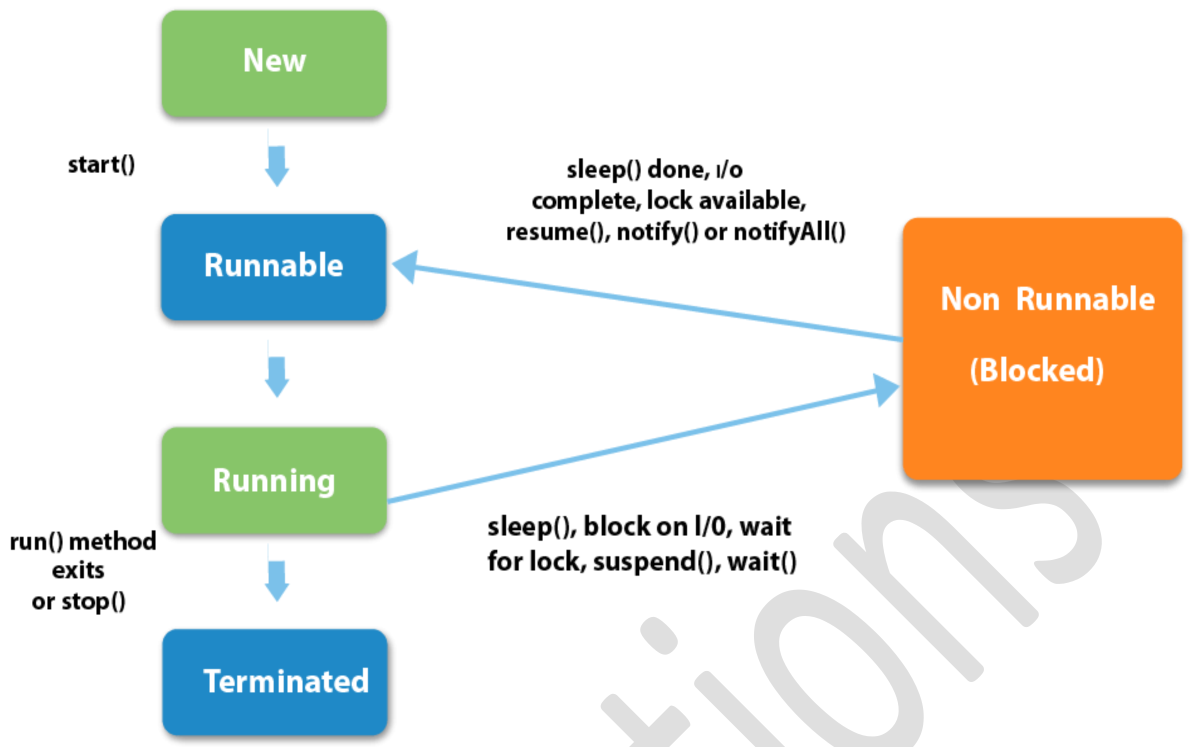
A thread can be in any of the five states:

1) **New:** When the instance of thread is created it will be in New state. Ex: Thread t= new Thread(); In the above example t is in new state. The thread is created but not in active state to make it active we need to call start () method on it.
2) **Runnable state:** A thread can be in the runnable state in either of the following two ways: a) When the start method is invoked or b) A thread can also be in runnable state after coming back from blocked or sleeping or waiting state.
3) **Running state:** If thread scheduler allocates cpu time, then the thread will be in running state.
4) **Waited /Blocking/Sleeping state:** In this state the thread can be made temporarily inactive for a short period of time.

Java Thread Methods

S.N.	Modifier and Type	Method	Description
1)	Void	start()	It is used to start the execution of the thread.

2)	Void	run()	It is used to do an action for a thread.
3)	static void	sleep()	It sleeps a thread for the specified amount of time.
4)	static Thread	currentThread()	It returns a reference to the currently executing thread object.
5)	Void	join()	It waits for a thread to die.
6)	Int	getPriority()	It returns the priority of the thread.
7)	Void	setPriority()	It changes the priority of the thread.
8)	String	getName()	It returns the name of the thread.
9)	Void	setName()	It changes the name of the thread.
10)	Long	getId()	It returns the id of the thread.
11)	Boolean	isAlive()	It tests if the thread is alive.
12)	static void	yield()	It causes the currently executing thread object to pause and allow other threads to execute temporarily.
13)	Void	suspend()	It is used to suspend the thread.
14)	Void	resume()	It is used to resume the suspended thread.
15)	Void	stop()	It is used to stop the thread.
16)	Void	destroy()	It is used to destroy the thread group and all of its subgroups.
31)	Void	notify()	It is used to give the notification for only one thread which is waiting for a particular object.
32)	Void	notifyAll()	It is used to give the notification to all waiting threads of a particular object.


New
start()
Runnable
sleep() done, I/o complete, lock available, resume(), notify() or notifyAll()
Non Runnable (Blocked)
Running
sleep(), block on I/0, wait for lock, suspend(), wait()
run() method exits or stop()
Terminated

Possible combination of method in Thread

1. wait() – notify(),notifyAll()
2. Suspend() – resume()

Difference between wait() and sleep()

- The wait() method can only be called from Synchronized context
- The wait() method releases the lock on an object and gives others chance to execute.
- The sleep() method does not releases the lock of an object for specified time or until interrupt.

Whenever a new **Java thread** is created it has the same **priority** as the **thread** which created it.

2 ways to create thread in Java

1. Extending Thread class

```
class Multi extends Thread{
public void run(){
  System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

2. By implementing Runnable interface

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}

public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
 }
}
```

Thread priorities

Type 1:

```
ThreadDemo t1 = new ThreadDemo();
ThreadDemo t2 = new ThreadDemo();
ThreadDemo t3 = new ThreadDemo();

System.out.println("t1 thread priority : " +
                   t1.getPriority()); // Default 5
System.out.println("t2 thread priority : " +
                   t2.getPriority()); // Default 5
System.out.println("t3 thread priority : " +
                   t3.getPriority()); // Default 5
 t1.setPriority(2);
t2.setPriority(5);
t3.setPriority(8);
```

Type 2:

```
TestMultiPriority1 m1=new TestMultiPriority1();
TestMultiPriority1 m2=new TestMultiPriority1();
TestMultiPriority1 m3=new TestMultiPriority1();
m1.setPriority(Thread.MIN_PRIORITY);
m2.setPriority(Thread.MAX_PRIORITY);
m3.setPriority(Thread.NORM_PRIORITY);
```

Type 3:

- All **Java threads** have a **priority** in the range 1-10.

- By default the main thread priority is 5.

```
  ThreadDemo t1 = new ThreadDemo();
  ThreadDemo t2 = new ThreadDemo();
  ThreadDemo t3 = new ThreadDemo();

System.out.println("t1 thread priority : " +
              t1.getPriority()); // Default 5
System.out.println("t2 thread priority : " +
              t2.getPriority()); // Default 5
System.out.println("t3 thread priority : " +
              t3.getPriority()); // Default 5
```

t1.setPriority(2);
t2.setPriority(5);
t3.setPriority(8);

Multitasking

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

- Process-based Multitasking (Multiprocessing)
- Thread-based Multitasking (Multithreading)

1) Process-based Multitasking (Multiprocessing)

- Each process has an address in memory. In other words, each process allocates a separate memory area.
- A process is heavyweight.
- Cost of communication between the process is high.
- Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.

2) Thread-based Multitasking (Multithreading)

- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.

Synchronization in Java

Synchronization in java is the capability *to control the access of multiple threads to any shared resource*.

Types of Synchronization

- Process Synchronization
- Thread Synchronization

Thread Synchronization

1. Mutual Exclusive
   - Synchronized method.
   - Synchronized block
   - .
2. Cooperation (Inter-thread communication in java)

Synchronized method

```
class Table {
        synchronized public void printTable(int n) {
    // method not synchronized
                for (int i = 1; i <= 5; i++) {
                        System.out.println(n * i);
                        try {
                                Thread.sleep(400);
                        } catch (Exception e) {
                                System.out.println(e);
                        }
                }
        }
}
class MyThread1 extends Thread{
Table t;
        MyThread1(Table t){
                this.t=t;
        }
        public void run(){
                t.printTable(5);
        }
}

public class Synchronization {

        public static void main(String[] args) {
                Table obj = new Table();//only one object
                MyThread1 t1=new MyThread1(obj);
                MyThread1 t2=new MyThread1 (obj);
        t1.start();
        t2.start();
    }
}
```

Synchronized block

```
class Table {
     public void printTable(int n) {// method not synchronized
         synchronized(this){
              for (int i = 1; i <= 5; i++) {
             System.out.println(n * i);
             try {
                 Thread.sleep(400);
             } catch (Exception e) {
                 System.out.println(e);
             }
         }
    }
}
}
class MyThread1 extends Thread{
Table t;
    MyThread1(Table t){
        this.t=t;
    }
    public void run(){
        t.printTable(5);
    }
}

public class synchronization {

    public static void main(String[] args) {
        Table obj = new Table();//only one object
        MyThread1 t1=new MyThread1(obj);
        MyThread1 t2=new MyThread1 (obj);
        t1.start();
        t2.start();
    }
}
```

Can we restart a dead thread in java?

If we try to restart a dead thread by using start method we will get run time exception since the thread is not alive.

Can one thread block the other thread?

No one thread cannot block the other thread in java. It can block the current thread that is running.

Can we restart a thread already started in java?

A thread can be started in java using start() method in java. If we call start method second time once it is started it will cause RunTimeException(IllegalThreadStateException). A runnable thread cannot be restarted.

What happens if we don't override run method ?

If we don't override run method .Then default implementation of Thread class run() method will be executed and hence the thread will never be in runnable state.

Can we overload run() method in java?

We can overload run method but Thread class start method will always cal run method with no arguments. But the overloaded method will not be called by start method we have to explicitly call this start() method.

Assume a thread has lock on it, calling sleep() method on that thread will release the lock?

Calling sleep () method on thread which has lock doesn't affect. Lock will not be released though the thread sleeps for a specified amount of time.

Can sleep () method causes another thread to sleep?

No sleep () method causes current thread to sleep not any other thread.

What are the benefits of multithreaded programming?

Multithreading enables to use idle time of cpu to another thread which results in faster execution of program. In single threaded environment each task has to be completed before proceeding to next task making cpu idle.

What is Exception?

- Events that occur during the execution of programs that disrupt the normal flow of instructions (e.g. divide by zero, array access out of bound, etc.
- Because of bad input or human error

What is exception hierarchy in java?

- Throwable is a parent class of all Exception classes.
- There are two types of Exceptions: Checked exceptions(or) compile time exception and Unchecked Exceptions or RunTime Exceptions.

What are the differences between Checked Exception and Unchecked Exception?

Checked Exception

- The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions.
- Checked exceptions are checked at compile-time.
- Example: IOException, SQLException etc.

Unchecked Exception

- The classes that extend RuntimeException are known as unchecked exceptions.
- Unchecked exceptions are not checked at compile-time.

What is default Exception handling in java?

When JVM detects exception causing code, it constructs a new exception handling object by including the following information.
1) Name of Exception
2) Description about the Exception
3) Location of Exception.

What are user defined exceptions?

- To create customized error messages we use userdefined exceptions.
- We can create user defined exceptions that extend Exception class or subclasses of checked exceptions. so that userdefined exception becomes checked.
- Userdefined exceptions can extend RuntimeException to create userdefined unchecked exceptions.

How to create a custom Exception?

To create you own exception extend the Exception class or any of its subclasses.

class New1Exception extends **Exception** { } // this will create Checked Exception

class NewException extends **NullPonterExcpetion** { } // this will create UnChecked exception

What is difference between Error and Exception?

Error:

- Bug in a **program,** irrecoverable condition occurring at run time**.** Such as OutOfMemory error.
- **errors** are abnormal conditions

Exception:

- Events that occur during the execution of programs that disrupt the normal flow of instructions (e.g. divide by zero, array access out of bound, etc. Because of bad input or human error

How can you handle Java exceptions?

There are five keywords used to handle exceptions in java:

- try
- catch
- finally
- throw
- throws

The possible combination of exception handling

- Try
- Try,catch
- Try,catch,finally
- Try,finally

What purpose does the keywords final, finally, and finalize fulfill?

Final: Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.

Code:

```
class Test
{
   static final int CAPACITY = 4;

   public static void main(String args[])
   {
      // re-assigning final variable
      // will throw compile-time error
      CAPACITY = 5;
   }
}
```

Output:

```
Compiler Error: cannot assign a value to final variable CAPACITY
```

Finally

Finally is used to place important code, it will be executed whether exception is handled or not. Let's take a look at the example below to understand it better.

Code:

```
class TestFinallyBlock{
  public static void main(String args[]){
  try{
   int data=25/5;
   System.out.println(data);
```

```
 }
 catch(NullPointerException e){System.out.println(e);}
 finally{System.out.println("finally block is always executed");}
 System.out.println("rest of the code...");
}
   }
```

Finalize

The java.lang.Object.finalize() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object. A subclass overrides the finalize method to dispose of system resourcesor to perform other cleanup.

Code:

```
import java.util.*;

publicclassObjectDemoextends GregorianCalendar {

publicstaticvoid main(String[] args) {
try {
        ObjectDemo cal = new ObjectDemo();

        // print current time
        System.out.println("" + cal.getTime());

        // finalize cal
        cal.finalize();
        System.out.println("Finalized.");

      } catch (Throwable ex) {
        ex.printStackTrace();
      }
   }
}
```

What are the differences between throw and throws?

throw keyword	**throws keyword**
Throw is used to explicitly throw an exception.	Throws is used to declare an exception.
Checked exceptions cannot be propagated with throw only.	Checked exception can be propagated with throws.
Throw is followed by an instance.	Throws is followed by class.
Throw is used within the method.	Throws is used with the method signature.
You cannot throw multiple exception	You can declare multiple exception e.g. public void method() throws IOException,SQLException.

Explain throw keyword in java?

The Java throw keyword is used to explicitly throw an exception.

- We can throw either checked or uncheked exception in java by throw keyword.
- The throw keyword is mainly used to throw custom exception.

Code:

```
class ThrowExcep
{
staticvoid fun()
    {
try
        {
Throw new NullPointerException("demo");
        }
catch(NullPointerException e)
        {
            System.out.println("Caught inside fun().");
throw e; // rethrowing the exception
        }
    }
Public static void main(String args[])
   {
try
      {
fun();
      }
catch(NullPointerException e)
      {
         System.out.println("Caught in main.");
      }
   }
}
```

Output:

```
Caught inside fun().

Caught in main.
```

Can we write any code after throw statement?

- After throw statement jvm stop execution and subsequent statements are not executed.
- If we try to write any statement after throw we do get compile time error saying unreachable code.

Explain importance of throws keyword in java?

- Now Checked Exception can be propagated (forwarded in call stack).
- It provides information to the caller of the method about the exception.

Explain the importance of finally over return statement?

- finally block is more important than return statement when both are present in a program.
- For example if there is any return statement present inside try or catch block , and finally block is also present first finally statement will be executed and then return statement will be considered

Explain a situation where finally block will not be executed?

Finally block will not be executed whenever jvm shutdowns. If we use system.exit(0) in try statement finally block if present will not be executed.

Explain the importance of throwable class and its methods?

Throwable class is the root class for Exceptions. All exceptions are derived from this throwable class. The two main subclasses of Throwable are Exception and Error. The three methods defined in throwable class are :

1) void printStackTrace() : This prints the exception information in the following format : Name of the exception, description followed by stack trace.
2) getMessage() This method prints only the description of Exception.
3) toString(): It prints the name and description of Exception.

Collection

What is collection?

- The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.
- All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion, etc. can be achieved by Java Collections

What are the benefits of Collections framework?

- Improves program quality and speed
- Increases the chances of reusability of software
- Decreases programming effort

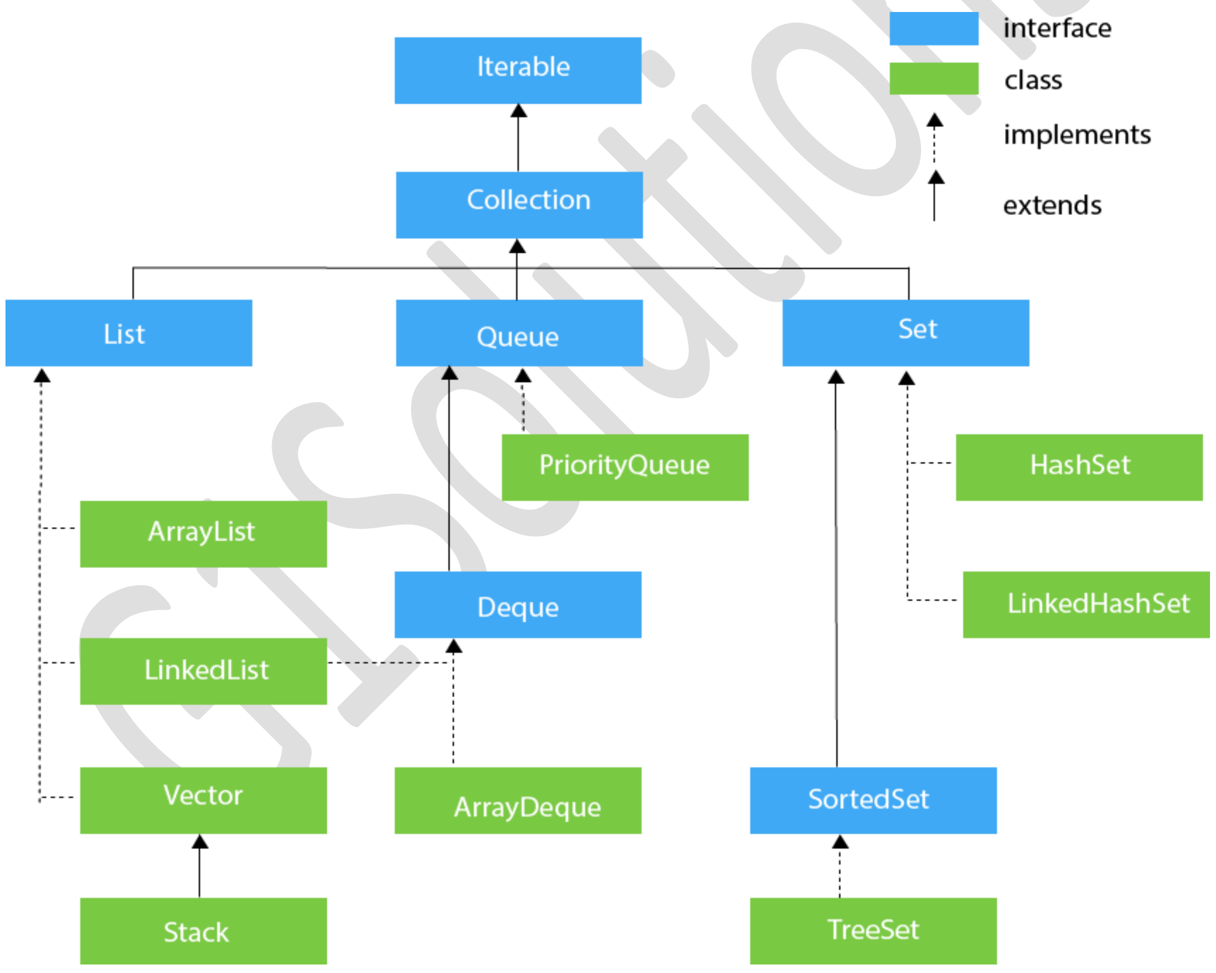
What is Iterator and its uses?

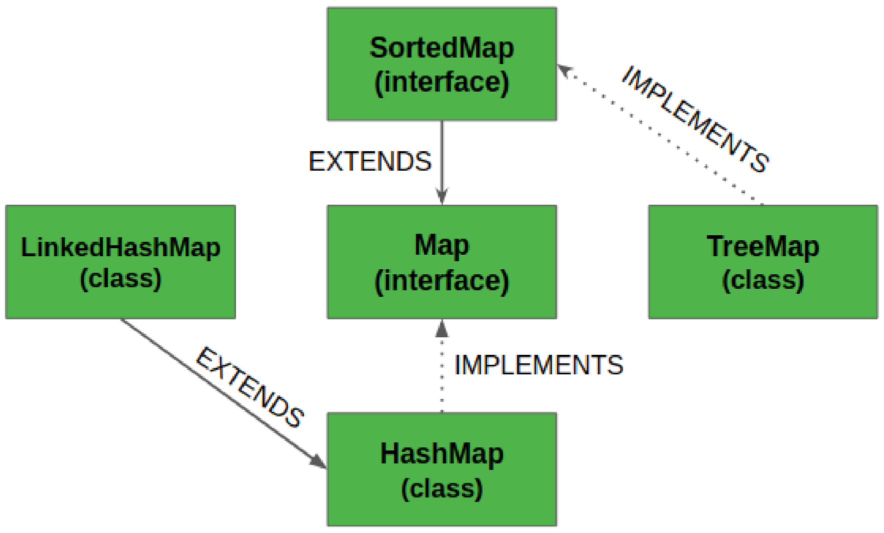
- Iterators are used in Collection framework in Java to retrieve elements one by one. There are three iterators.
- By using Iterator, we can perform both read and remove operations.
- Iterator object can be created by calling iterator () method present in Collection interface.

List Interface:

- List interface is the child interface of Collection interface.
- List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

```
List <data-type> list1= new ArrayList();
List <data-type> list2 = new LinkedList();
List <data-type> list3 = new Vector();
List <data-type> list4 = new Stack();
```





MAP Hierarchy in Java

	Insertion Order	*Duplicate*	*Null*	*Synchronized*
Array List	✓	✓	✓	×
Linked list	✓	✓	✓	×
Vector list	✓	✓	✓	✓
HashSet	×	×	✓	×
Linked Hashset	✓	×	✓	×
Treeset	×	×	×	×
Hashmap	×	×	✓ 1K-NV	×

Linked Hashmap	✓	×	✓ 1K-NV	×
Treemap	×	×	×	×

```
ArrayList<String> list=new ArrayList<String>();//Creating arraylist
list.add("Ravi");//Adding object in arraylist
list.add("Vijay");
list.add("Ravi");
list.add("Ajay");
//Traversing list through Iterator
Iterator itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
```

Set Interface

1. Set<data-type> s1 = new HashSet<data-type>();
2. Set<data-type> s2 = new LinkedHashSet<data-type>();
3. Set<data-type> s3 = new TreeSet<data-type>();

```
    Set<String> set=new TreeSet<String>();
    set.add("Ravi");
    set.add("Vijay");
    set.add("Ravi");
    set.add("Ajay");
    Iterator itr=set.iterator();
    while(itr.hasNext()){
    System.out.println(itr.next());
    }
}
```

```
Iterator itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
```

Vector

Code:

```
import java.util.*;
class Vector_demo     {
Public static void main(String[] args)  {
       // create default vector
       Vector v = new Vector();
       v.add(0, 1);
       v.add(1, 2);
       v.add(2, "vijay");
       v.add(3, "Ajith");
       v.add(4, 3);

       System.out.println("Vector is " + v);

   } }
```

Define Linked List and its features with signature ?

Linked list is used for storing a collection of objects that allows efficient addition and removal of elements in the middle of the collection.

6 Methods in linked list?

Important methods specific to LinkedList class:

1) public E getFirst() : getFirst() will returns the first element in the list.

2) public E getLast() : getLast() returns the last element in the list.

3) public E removeFirst() : removeFirst() method removes the first element in the list.

4) public E removeLast() : 40 removeLast() method removes the last element in the list.

5) public void addFirst(E e) Inserts the element at beginning of the list.

6) public void addLast(E e) : Inserts the element at end of the list.

Code:

```
import java.util.LinkedList;

import java.util.List;

public class LinkedListGenericsDemo

{

  public static void main(String[] args)

  {

          List<String> names = new LinkedList<>();

          names.add("Rams");

          names.add("Posa");

          names.add("Chinni");

        // We cannot add other than Strings

        // names.add(2011);
        System.out.println("LinkedList content: " + names);

        System.out.println("LinkedList size: " + names.size());

  }

}
```

Map interface in java?

A map is an association of key-value pairs. Both keys and values in map are objects. Features of map: 1) Maps cannot have duplicate keys but can have duplicate value objects.

Code:

```
import java.util.*;
class HashMapDemo
{
   public static void main(String args[])
   {
      Map< String,Integer> hm =new HashMap< String,Integer>();
      hm.put("a", new Integer(100));
      hm.put("b", new Integer(200));
      hm.put("c", new Integer(300));
      hm.put("d", new Integer(400));

      // Returns Set view
      Set< Map.Entry< String,Integer>> st = hm.entrySet();

      for (Map.Entry< String,Integer> me:st)
      {
         System.out.print(me.getKey()+":");
         System.out.println(me.getValue());
      }
   }
}
```

What is SortedMap interface?

SortedMap extends Map interface.Sorted Map maintains sorted order of keys in a map. By default sorted map maintains natural ordering if we want custom order we can specify using comparator. public interface SortedMap extends Map { }

Code:

```
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.TreeMap;

publicclassSortedMapExample
{
publicstaticvoid main(String[] args)
    {
        SortedMap<Integer, String> sm = new TreeMap<Integer, String>();
        sm.put(new Integer(2), "practice");
        sm.put(new Integer(3), "quiz");
        sm.put(new Integer(5), "code");
        sm.put(new Integer(4), "contribute");
        sm.put(new Integer(1), "geeksforgeeks");
       Set s = sm.entrySet();
        // Using iterator in SortedMap
Iterator i = s.iterator();
        // Traversing map. Note that the traversal
        // produced sorted (by keys) output .
while (i.hasNext()){
Map.Entry m = (Map.Entry)i.next();

int key = (Integer)m.getKey();
            String value = (String)m.getValue();

            System.out.println("Key : " + key +
                        "  value : " + value);
        }
    }
}
```

What is Hashtable and explain features of Hashtable?

- Hashtable was available before collection framework.
- Hashtable offers a convenient way of storing key/ value pairs.
- Hashtable does not allow nulls either keys or values.
- Hashtable is synchronized.

Code:

```
import java.util.*;
class hashTabledemo
{
   public static void main(String[] arg)
   {
        //creating a hash table
      Hashtable h = new Hashtable()
      Hashtable h1 = new Hashtable();

      h.put(3, "vijay");
      h.put(2, "dinesh");
      h.put(1, "kumar");

      // create a clone or shallow copy of hash table h
      h1 = (Hashtable)h.clone();

      // checking clone h1
      System.out.println("values in clone: " + h1);

      // clear hash table h
      h.clear();

      // checking hash table h
      System.out.println("after clearing: " + h);
     }
}
```

Difference between Collection and Collections in java?

Collection: represent group of objects where objects are stored. This is one of the core interface which provides basic functionality for collection.

Collections: Collections contains some utility static methods that operate on collections.

Difference between HashMap and Hashtable?

HashMap	**Hashtable**
HashMap is not synchronized.	Hashtable is synchronized.
Nulls HashMap allows atmost one null key and any number of null values.	Hashtable does not allow null values.
Performance Since HashMap is not synchronized its performance is faster than Hashtable.	Performance is slower when compared to HashMap. Introduction HashMap introduced starting from Hashtable is even before collection

Difference between arraylist and linkedlist?

Difference	**Arraylist**	**LinkedList**
Access	Implements RandomAccess interface we can search randomly all the elements in the list.	It extends Abstract sequential List interface which provides sequential access to elements.
Searching and retrieval of elements	Searching and retrieval of elements is fast since arraylist provides random access	Searching and retrieval of elements is slow because of sequential access to elements.

Addition and removal of elements	Adding and removal of elements in random positions is slow.For example if we want to add element to middle of the list we have to move the elements in the list and then we need to insert the element. Similarly for removing the element we need to follow the same thing.	Adding and removal of elements in random positions is fast because there is no need of resizing the array just by updating the node structures with new addresses

Stream using Collection List to map:

Example:

We use stream in Java to convert given list to stream, then stream to set. This works only in Java 8 or versions after that.

```
// Java program to demonstrate conversion of
// Set to list using stream

import java.util.*;
import java.util.stream.*;

class Test {
    public static void main(String[] args){

        // Creating a list of strings
        List<String> aList = Arrays.asList("vijay", "hi",
                    "hello", "bye", "Goood");

        // Converting to set using stream
        Set<String> set = aList.stream().collect(Collectors.toSet());

        for (String x : set)
            System.out.println(x);
    }
}
```

Using Collectors.toMap() method: This method includes creation of a list of the student objects, and uses Collectors.toMap() to convert it into a Map

List to Map Convertion using stream:

```
  public class GFG {

      // main Driver
      public static void main(String[] args) {
        // create a list
       List<Student> lt = Arrays.asList("vijay", " Ajith ", "kumar");
                    // create map with the help of
                    // Collectors.toMap() method
  LinkedHashMap<Integer, String> map = lt.stream().collect(
  Collectors.toMap(Student::getId, Student::getName, (x, y) -> x + ", " + y,
  LinkedHashMap::new));
  // print map
  map.forEach((x, y) -> System.out.println(x + "=" + y));
         }
       }
```

Web Application using springboot webservices:

Project Flow:

Project creation in sts:

File ->new -> Spring Starter Project

➔ **We choose project needed Dependencies :**

- **MySQL Driver**
- **SpringbootDevTools**
- **Spring Data JPA**
- **Spring Web**

In my project we are working in spring boot Application. Here we have 1 module and 4 layers. In those layers like Controller, Service, DAO and Repository

In Entity Class we are using **@Entity** annotation it is used to mention this is the Entity class. After we use **@Table** annotation its used for creating the table in database.

Here after we using **@Id** annotation it is used for specifies the primary key of an entity and the **@GeneratedValue** provides for the specification of generation strategies for the values of primary keys

Example:

```
@Entity
@Table(name = "producttable")
public class Product{

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name = "id")
        private int productId;
        @Column(name = "name")
        private String Name;
        @Column(name = "price")
        private double price;
```

Step:1

http://localhost:8080/product/products

Hit that url via postman client sending the request to controller class

Step:2

Request to Controller class:

In Controller class we are using **@RestController** annotation its used to mention this is the controller class.

After that we using **@RequestMapping** it used to map the HTTP request from URL to controller class.

After that we using **@autowired** annotation its used for dependency Injection to create the object in the service class.

Here we are using **CRUD** operation like Create, Read, Update and Delete. Here after we using HTTP methods like Postmapping, Putmapping, getmapping and deletemapping.

Here we using **@Pathvariable** annotation its extract the value from the URL. and depends upon we use **@RequestBody** Annotation it's also extract the object value from the URL.

Example:

```
@RestController
@RequestMapping("/product")
public class ProductController {

        @Autowired
        private ProductService productService;

        @RequestMapping(value = "/addproducts", method =
RequestMethod.POST)
        public ResponseEntity<List<Product>> createProduct(@RequestBody
Product product) throws CustomerException {

                Product product = productService. saveProduct(Product product);

           //save the database. if the product is null its throws the
CustomerException otherwise its throws success response in postman

          if (product == null) {
                return new ResponseEntity<>(product,
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                return new ResponseEntity<>(product, HttpStatus.OK);
                }

      @GetMapping("/product")
        public List<Product> findAllProducts() {
                return productService.getProducts();
                }

      @PostMapping("/addProduct")
        public Product addProduct(@RequestBody Product product) {
                return productService.saveProduct(product);
                }

      @PutMapping("/update")
        public Product updateProduct(@RequestBody Product product) {
                return productService.updateProduct(product);
        }
        @DeleteMapping("/delete/{id}")
        public String deleteProduct(@PathVariable int id) {
                return productService.deleteProduct(id);
                }
```

Step:3

After Request to Service Class:

In this class we using **@Service** annotation it’s have Business logic in class level.

In this interface we use **@Repositary** Annotation its used for access to Data Base directly in class level.

Here the **@autowired** annotation was used to dependency injection in repository. Here we using some predefined methods like save, find, delete.

Here we do Business Logic in service class

Example:

```
@Service
public class ProductService {

        @Autowired
        private ProductRepository repository;

      @Override
        public Product saveProduct(Product product) throws CustomerException {
          try {
          Product existingProduct = new Product();
                existingProduct.setName(product.getName());
                existingProduct.setQuantity(product.getQuantity());
                existingProduct.setPrice(product.getPrice());

     } catch (Exception e) {
                        throw new CustomerException("Error", e.getMessage());
        }
                return repository.save(product);
        }
```

Step:4

After that Request to Repositary Interface by using Extends keyword JpaRepository:

Here Repository was extending with JPA repository. we using extends keyword for extends a class. In JPA repository have some query and several predefined program. Query and some predefined methods are use request to do it. This is called serialization. Here text format convert to JSON format.

After that response happening. Its repository to DAO it will after that service to controller here controller to UI its called deserialization. Here JSON format convert to text format happened. We are getting response in post client

Example:

```
public interface ProductRepository extends JpaRepository<Product, Integer>{

    List<Product> save();

    @Query("Select q from product q ORDER BY id,name,price")
      Product findByName(String name);
}
```

Application Properties:

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/product
spring.datasource.username = root
spring.datasource.password = root
spring.jpa.show-sql = true
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
server.port=8080
```

How to change the port number?

Change the port number by using **application properties**.

Example:

```
server.port=8080
```

Repository: GitLab:

Local repository code management

- Gitpush: Code changes using push local repository
- Gitpull: Someone update the latest code pull newest
- git merge
- gitlog –summary detailed view changes

Application Server:

wildfly