JAVA

- team of 4 members (James Gosling)

-Cable TV problem. - TV with set top box(Digital TV)

OAK - JAVA(island name) - coffee mug. reg under Sun Microsystems pvt LTD.

Microsoft Takes Over Java from 1994 till 2010.
From 2010 till date officially Sun microsys is part of Oracle Co-op.
Latest Version - Java 13. Sept 2019.
Official Site - www.oracle.com

Over 3.5 billion devices run on java.
Andriod, Set top Boxes, Mobile devices, Symbian OS(JME), Portals where security is involved.

1)Console App
2)Stand Alone APP
3)Enterprise or Distributed or Web Apps
4)Micro Apps

JSE - Java's Standard Edition (Core Java) with some Advanced Stuff too.
Core Java - OOP using Java, Programming Fundamentals using Java.
Advanced java - MultiThreading, Error or Exception Handling
, Collections F/W, JDBC, Swings & Applet(not covered)
JEE - Java's Enterprise Edition(Java on Web or at server-side)
Servlet, JSP, EJB, RMI, JSF.
JME - Java's Micro Edition(Java on a micro platform)

Some open source f/ws
- Hibernate , Spring , Struts, Portlets

-SOA, Web services using SOAP and REST.

Features or Props of Java

---

1) Java is Platform Independent. WORA.
2) Java comes up with garbage collected by default.
3) Java is compiled and interpreted language.
4) Java has the 2nd fastest compiler embedded in him, where the best compiler design techniques using Theory of computation is been implemented whose name is JIT(Just in time) compiler.
5) Java seprates its huge classification of predefined classes into smaller folders called as packages.
6) Java has strongly built all the 5 principles of Object Orientation and it is the purest form of Object Oriented Programming.
-since it is strongly built on all OOC thats what makes java highly secured.
7)java seprates its developers stuff into a packaged environment called as JDK(Java development kit) and to execute any of the code the developer has written he makes use of JRE(Java Runtime Environment).
8)java is simple, developer friendly, portable, highly secured and more.
9)java comes up with bottom to top approach which eases the way of writing the code.
10)Using java you can develop multithreaded(parallel programming) apps.

OOC

---

1)Class & Objects
2)Encapsulation
3)Inheritance (Code Reusability)
4)Polymorphism
5)Abstraction

access specifiers or visbility modes

---

public -> protected -> default -> private

1)Classes & Objects

---

* a class is a reflection or a template or a blueprint of any real world object which has a state and a behaviour.

* a class can contain global variables or objects(instance or object & class level(static)),
static and non static methods, Constructors, inner classes.
* a class can either be public or default and if it is public then the class name and the file name should and must be same which also in other words a .java file can have only and only one public class for which the .java file name and the class name should and must be same.
There can be any number of default classes in a .java file but there will be only and only one public class with the class name and file name to be absolutely same.
* business or best industry practices tells about keeping only a single class per .java file and highly recommends to use public.

static keyword
---
*static are the things which are loaded directly inside the Stack.
* such things can be accessed directly without creating any object.
* some reusable or frequently accessed stuff will be made as static.
* static stuff are considered to be as class level.
* static also refers to, there will be only and only one such copy.

What is object?
---
object is a instance of a class.
you create the objects to access the non static content of a class.
Classname objname=new Constructorname();

Constructors
---
* Constructor is a special method of any class with the same name as the class name but without any return type.
*Constructors are used to intialize the environment for any class , i.e whenever an object is getting created for a particular class thats when the constructor of that class is invoked and it can help us setup the environment for that particular class.

ClassName obj=new ConstructorName();

*the java compiler creates
a default(visibility) non parameterised empty constructor in all the classes which it will take off when a pragrammer creates one of his own constructor.

```
* as per the best coding practices its highly recommended to have a
public non parametrised empty or non empty constructor in all the classes.

* a class can have any number of constructors but with different argument or
parameter list. such thing is called as constructor overloading.


this keyword
------------------
-> this is used to point the current class objects(global instance variables)
-> this can also be used in chaining of constructors in same class at the first
line of
constructor only.

Encapsulation
---------------------
Encapsulation talks about binding up the instance variables of a class with its
methods tightly so that the instance variables wont be accessd
directly with a object. To set the value for such object the programmer
should make use of a construtor or a appropriate setter method, and to
retrive or read the value he should make use of appropriate getter method.
This can be acheived by making the instance variables as private and
keeping appropraite setter and getter for the same.
for eg:
public class Student{
private int studentId;
private String studentName;
private double marks;

public void setStudentId(int studentId){
this.studentId=studentId;
}
public void setStudentName(String studentName){
this.studentName=studentName;
}

public int getStudentId(){
return studentId;
}
public String getStudentName(){
return studentName;
}
```

```
public Student(){
}
}

-- Such class with private types and with appropriate setters and getters
is called as encapsulated class or bean class or POJO(plain old java object)
class or VO(Value Object) class or DTO(Data transfer Object) class or Model class.



Inheritance(Code-Reusability)
---------------------------------
*Inheritance talks about the code reusability.
where the intial class for solving a particular problem which we
create is considered to be as a parent which would've given a solution
to existing problem, now when new problem statement raises we will create
a Child class for this parent and give the additional benifits in it.

*So child gets all the properties of parent apart from constructors and
private stuff.

*Such derivation between a parent or base or super class and a child
or derived or sub class is basically called as inheritance where the property
which has been written in any of parent(ancestors) will be given to
child where child doesnt have to rewrite it but it has a power to modify it
which will be covered in polymorphism.
* Basically you can create the object for a child class and access all the
properties of a chils as well as its ancestors.
* in java by default all the classes are subclass. i.e all the
classes in java has a parent by default who is java.lang.Object which
is super class of all the java classes.
* use extends keyword to bring in a relationship between a parent and
a child like below
class Child extends Parent{}
*whenever we create a object for a child jvm internally calls up
all the default constructors of its parents(ancestors) to identify the
methods in it.
*remember that using the child class object u can access all the props
of parent but a parent cannot access any properties of its child.

*there are types of inheritance like
-> single level (one parent and one child)
```

-> multi level (grandparent -> parent and a -> child)
-> hierarchical (a father has 4 kids and those kids have 2 kids)family tree.
-> multiple (father,mother has a child) not supported in java due to daimond problem.
-> hybrid(combination of multiple and hierarchical)not supported

what is daimond problem?
it occurs in multiple inheritance where Child has 2 immediate or same level parents like Father class and Mother Class.
if in this the Father class and the Mother class has same method with same param list then the child will be in a ambigous state to which method needs to be called or invoked either Father's or Mother's.
such ambigious situation is called as daimond problem.

super keyword
---
super always point to immediate parent class object,
using which u can call the constructors or public members of immediate parent.
this is like chaining the child class constructor with parent which should again be done in a first line of any child class constructor.

4)Polymorphism(poly-many, morphs - forms)
---
a)CompileTimePolymorphism(static binding) -> here the objects are binded at compile time
to give the sam name for the core functionality with different set of param list which is called as method overloading.
b)RunTimePolymorphism(dynamic binding) -> here the objects are binded at the run time, and
mainly this type of polymorph happens between a parent and a child where the props of parent is modified inside a child, the method signature including the param list will be same in a parent and a child, where when the creation of object is happening at a runtime, it will encounter with 2methods with same name and arg list in which it will consider the Child method and ignores the parent method, this is what is called as method overriding.

Object Casting

```
-------------------
where a child class object will be converted to parent and vice versa.
implicit object casting -> where child class object is getting assigned to
                      parent class object.
explicit object casting -> refers to parent to child conversion which is
               explicitly done by a programmer.



Parent p=new Parent();
Child c=(Child)p; // this is not casting ... ClassCastException


Parent p=new Child(); // Implicit
Child c=(Child)p; // Explicit




final keyword
--------------------
* can be used at 3 levels/places
*class level -> such classes cannot have child class.
*method level -> such methods cannot be modified or cannot be overridden.
* at variable or obj level -> it is called as constants which cannot be
                      modified at all.
static final int X=10;
x++; // will throw me an error as final variables are non modifiable.



5)Abstraction
-----------------------------
abstarct data refers to that data whose implementation details
are hidden.
the customer or end user is never intrested in how the features are
implemented, he will be intrested in weather the feature is given to him
or not, hiding up of such implementation details is what is called as abstraction.

abstraction in java can be acheived or done in 2 ways
first way -> by creating abstract classes (partial abstraction)
second way -> by creating interfaces (pure form of abstraction)
```

-> acheiving abstraction by creating abstract classes (partial abstraction)
---
* a class is said to be an abstract class if it is having at least one abstract method.
what is abstract method? -> abstract method is a method for which the implmentation or body of method wont be there.
where is the implemenation of abstract method? -> the abstract method is implemented or defined in the child class or the class which is extending it.
* abstract classes can have both abstract as well as non abstract methods.
* abstract classes cannot be instantiated i.e u cant create a object of a abstract class.
* whichever class extends such abstract class will be having 2 options which is
   -> implement all the abstract methods of its parent
   -> else make it also as abstract.
* abstract classes or methods cannot be static or final.
* apart from this it is just like a class.

-> acheiving abstraction by creating interfaces(pure form of abstraction)
---
*interfaces are also just like a abstract classes itself but they wont allow any sort of non abstract methods in it.
*whatever u write within a interface is by default abstract, infact you dont have to use abstract keyword to the method at all.
*interface in java solves the daimond problem.
* all the methods in interface are by default abstract as well as public.
* if you declare any variable within a interface then that variable is considered to be as a final variable.
*a class can be extended only once but an interface can be implemented any number of times.
* apart from this it gets all the properties like it can have inner interface, anonymous interface, and you can also extend a interface by other interface.
* use implements keyword to implement any interface.
there are types of interfaces
1)Normal Interface -> interface with many abstract methods in it.
                      Eg : Collection, List, Set, Queue, Map
2)Functional Interface(Functional Programming)(SAM interfaces) -> Interface with only and only
                      one abstract method.

Eg : Runnable, Comparable, Comparator
@FunctionalInterface(from Java 8)

3)Marker Interface(Abstract Interfaces) -> Interface with no abstract methods at all, which
is used for compiler to mark that it does some job.
Eg: Serializable, Clonable

Wrapper Classes
---
*here the primitives are wrapped into a object to acheive
pure form of OO.
*collection f/w accepts only object so if i want to store any integers
or primitives then the wrappers are solution for that.
* all the wrappers are part of java.lang package.
* so the primitives like byte,short,int,long,float,double,char and boolean
are converted to Byte,Short,Integer,Long,Float,Double,Character,Boolean.
*since it is a class it comes up with lot many built in functions to make
programming easier.

1)String class
---
* java reads everything as a String and writes everything as a String.
* any object in java can easily be converted to String object
by simply calling toString() which is there in our super class
java.lang.Object.
*String is a final class.
*String is a immutable class i.e String objects are non - modifiable.if
you try to modify the String object it will go ahead and create a new
memory in the String pool.
* anything u write within "" is considered as a String object in java.

all the 9 classes (8 wrappers and one String class) overrides equals() and
toString()
method from Object class to compare the values and print it respectively.
and all these classes also do implement Comparable and Comparator interfaces
using which the java's inbuilt API will help in sorting the Objects.

2)StringBuffer and StringBuilder
- these are mutable classes
- there wont be any common pool for these objects you need to create
these using new keyword only.

- StringBuffer is thread safe or synchronized i.e it can be used in inter-thread communication where as StringBuilder is not thread safe or not synch.

Exception Handling

What is error? What is Exception?
*error is a non recoverable situation.
*exception is something which can be handled and can continue the same flow of execution.
* basically in any application due to user's input or due to our logical error it may lead to abnormal termination of a application which is a very bad practise, an application which terminates abnormally is considered as 0 quality app and nobody wishes to buy or get such application.
*Exception handling will help you handle the exceptions which has been raised and terminate normally with the proper communication to the end user, where such app which handles all the abnormal status normally its the app which is sold highest in any market.
*It is always encouraged to write a better exceptional handling to get a better review or quality about the product.
* There are set of predefined exceptions and there can be created some of userdefined or custom exceptions for fulfilling business requirement too.
* All the predefined Exceptions comes under Throwable class.
* and java.lang.Exception is a super class for all the exceptions in java.
*There are 2 types of exceptions
  a)Checked Exception(java.lang.Exception) -> this exception forces the progarmmer to handle it
                    at the compile time itself, until the programmer
                    handles it, compiler wont allow to run.
                    some of Checked exceptions are
                    FileNotFoundException, ClassNotFoundException,
                    MethodNotFoundException, SQLException and more.
  b)Unchecked Exceptions(java.lang.RuntimeException) -> These exceptions are part of java.lang.RunTimeException
                       which occurs at a run time, it is upto programmer
                    weather he wants to handle this or not, if he
                    doesnt handle it, it will lead to abnormal termination.

few unchecked exceptions are:
ArithmeticException, NullPointerException,
ConcurrentModificationException,
ArrayIndexOutOfBoundException,
StringIndexOutOfBoundException and more.

Exceptions can be handled using:
1)try...catch...finally
2)throws
3)throw

1)try...catch...finally
---
* try is a block of code where we keep the code which is suspected to generate a exception.
*catch is a handler which handles the exception raised by try block.
*finally is such block of code which is executed despite of exception is caught or not, we can use such block to clear our resources.
*a try block should and must be followed by catch or finally, u cannot write try alone.
*there can be any number of catch blocks for a single try which should be written in the order i.e the child class exception should come first and followed by the rest.
*finally and try will be a single block.
*if no catch you can even write finally, if there is a catch then finally comes at the end of all catch blocks.

2)throws
---
used to throw multpile exceptions at the method level to the calling method.
use it to force the programmer who is creating the object for that method will handle it.

3)throw
---
used to create the exception object of our own to throw our own customised exception with our own customised message.

Collection F/w
---
1)all the collections are resizable.

2)all the collections are dynamic.
3)Lots of built in functions to provide the ease of performing CRUD, search n sort.
4)All the collections accept only and only Objects.
5)all the collections are Iterable.
6)The root interface for all the collections except Map is Collection.
- under Collection interface we get List, Set , Queue and many more interfaces.
- List, Set and Map are widely used collections even in advanced f/ws like hibernate and spring you get to see lot on these things.
7)all the collections are part of java.util package
8)we get to see a utility class called as java.util.Collections in which you can find sort, search and many other functions which can be applied on a Collection.

List<E> Interface
---
*List looks exactly like an Array but it is dynamic.
*maintains order of insertion.
*allows duplications.
*allows any number of null values.
*List is implemented in ArrayList,Vector and LinkedList classes.
*List is indexed i.e it has positions for all the values.

ArrayList<E> class
---
*Gets all the props of List Interface.
*It isnt thread safe.
*by defualt 10 memory locations once filled, it will regrow by doing this -> previous_memory_size+ 50% of previous_memory_size.
*Contiguous

Vector<E> class
---
*Gets all props of List.
* it is thread safe class.
*by default 10 memory locations once it is filled it will give 10 more aand goes on.
*Contiguous

LinkedList<E> class
---
*LinkedList implements List,Queue and Dequeue.
*It isnt thread safe class.
* It allocates only one memory at a time when new element is coming in

it will create one more.
*Memory is not contiguous.
*Insertion and Deletion is much faster here then other classes.

Set<E> interface
---
*Set is a collection which is unindexed(no positions for elements)
and it wont allow dups.
*Set internally uses hashing algorithm which generates the memory
based on the input data, hence it wont allow dups.
*internally it checks for duplications using equals & hashCode method.
* Set is implemented in HashSet,LinkedHashSet,TreeSet classes.
*memory by default 16 allocations and load factor is .75

HashSet<E> class
---
* HashSet gets all the properties of Set.
*It is unordered i.e it wont preserve the order of insertion.
*Its not thread safe.
*allows one null.

LinkedHashSet<E> class

It is a subclass of HashSet
only difference is it uses DoublyLinkedList internally to preserve the
order of insertion.

TreeSet<E> class
---
*It implements Set, SortedSet and NavigableSet.
*By default using Comparable object it will maintain the data
in sorted order
*No null allowed here.
*Its faster than other Set's since its maintaing the data in sorted.
*not thread safe

Map<K,V> interface
---
*Map is a part of Collections f/w but not under Collection interface.
*Map comes with Key and Value pair where every Value is dependent on the Key.
*Key is unique and value can be duplicated.
* it is unindexed an unordered.

*Map is implemented in HashMap,LinkeHashMap,TreeMap and Hashtable classes.

HashMap<K,V> class
---
*Gets all the props of Map.
*one key can be null and any number of values can be null.
*doesnt maintain the order of insertion.
*not thread safe.

LinkedHashMap<K,V> class
---
*Its child of HashMap, so it gets all the props of Map and HashMap.
*Only difference is it maintains the order of insertion.

TreeMap<K,V> class
---
*It implements Map, SortedMap and NavigableMap interfaces.
*not synch
* maintains the order of a key by using Comparable/Comparator object.
* key cant be null, any number of values can be null.
* since its sorted its faster for searching n traversing.

Hashtable<K,V> class
---
*It implements Map.
* generally known as no null Map or no null table i.e neither a key nor
  a value can be null over here.
*it is thread safe.
* it wont maintain the order of insertion.

MultiThreading
---
* set of instructions is a program -> any program under exceution is a process
  -> part of this process or tiny process or smallest job of a process
  is basically called as thread. group of these threads sharing the load of
 process and doing some job parallelly in an application, such app
 we call it as multithreaded app.
*thread basically will help u acheive parallel programming i.e simultaneously
u can do multiple jobs and make your app much smarter and much faster.
remember the app which is faster is appreciated in the market and we have
more customers or clients basically for such app.
* for eg take a MS Word which is actually multithreaded app. so msword is

a process which gets created when u double click on it, so now ur working on this word doc u do a spelling mistake, for this u have a thread within a word app which highlights your mistake in red color so that u can go ahead and correct your spelling, and parallely if u have missed ur grammer green line u can see coming up between the text asking u to fix your sentence with proper grammer.

* by default all the java app is single threaded, i.e main is a application thread created by java.

* when u create a group of threads they are in a race condition to run and finish their jobs, in which u cant identify which thread is coming first and which is last.

* whenever a thread gets created in java it will be having 3 things with it [thread_name,priority,thread_group] [main,5,main]

priority by default will be 5 or it depends on the thread which is creating this thread, which ever thread is creating a thread the priority of that particular thread will be passed to new thread.

MIN_PRIORITY - 1

NORM_PRIORITY - 5

MAX_PRIORITY - 10

*thread has a life cycle -> you have to create a thread i.e you have to give a name, priority an group for a thread, next phase it will be put into runnable phase where thread is been checked weather it has permission to access or not, when thread starts to do its job we call thread is in running phase, in this running phase it has interrupts which may be generated to wait or to sleep or if some thread with higher priority is coming in, once thread completes its running phase it dies.

*remember thread will be given the memory which has been allocated to process and every thread will be having its own private stack to maintain the data of the particular thread.

*there are basically 2 types of thread

a)application/foreground thread - these threads which are designed to manage the foreground operations, the jvm will wait for all these threads to complete the job and then only it will exit.

b)deamon/background thread - these threads run in a background of our app mostly desgined for releasing resources or memory, jvm will never wait for any background thread to finish the job. if all the foreground threads are done with the job jvm will exit no matter at that time any background thread is running or not. for eg : garbage collector.

*in java, threads can be implemented in 2 ways

a)by extending java.lang.Thread class - where you have all the thread life cycle based methods.

b)by implementing java.lang.Runnable interface(preffered) - where you have only

run() method in which you will give the
job of thread.
* the threads default behaviour of being in a race condition wont be helpful when there are commonly shared resources, so you can build a synchronized block or a method where you can keep the shared data, in this block or a method only one thread is allowed at a time, until that thread releases the lock, other thread cant acquire that particular block.
*every thread in a java will be having a unique ID, using which jvm tracks the behaviour of it, all the status and behaviours are being managed using ThreadMonitors in java.
*start() is a method using which we can start a particular thread which will call the run() method. in run() method is what you will be giving the job of what exactly that thread should do.

File

InputStream OutputStream
**ByteStream(does reading nd writing byte by byte) - InputStream->FileInputStream, BufferedInputStream
OutputStream->FileOutputStream, BufferedOutputStream
In bytestream whenever u wanna write data to file u have to convert ur data to bytes and then u have to write it to the destination.
While reading also the stream gives u byte. convert in into char to get your desired
output.
**CharacterStream(does reading and writing caharacter by character)
- Reader -> FileReader,BufferedReader
- Writer -> FileWriter, BufferedWriter
In char stream no need to convert it while u read and write.

Note: Use Buffer in both the Byte and Char streams to speed up the reading and writing process.

Serialization
---
serializing and de-serializing a particular object/s in some persistant area.
Serilizable-- Interface

```
ObjectOutputStream - writeObject
ObjectInputStream - readObject
transient(use this if u dont want to serialize a particular object in a serialized
class)

JDBC
------
*Java to Database Connection
*JDBC is a driver which helps Java application to communicate with any SQL/
PLSQL provider.
*It has 4 types of Drivers
Type1 -  JDBC-ODBC Bridge Driver
Type2 - Java Native API Driver
Type3 - Java Network Protocol Driver - Pure Java Driver.
Type4 - Database Protocol Driver - 100% pure java driver.(thin driver)

*Steps to connect to database
1) Load or Register your Driver Details.
2) Open Connection (url, username, password)
3) Create Statement
4) Execute Statement ( boolean execute() prefferably DDL,
                      int executeUpdate() prefferably DML,
                         ResultSet executeQuery() prefferably DQL )
5)Process the Results
6)Close Connection

mysql details
url - jdbc:mysql://localhost:3306/database_name
username - root
password - root
drivername - com.mysql.jdbc.Driver

there are 3 types of JDBC Statements
java.sql.Statement - > everytime u pass your SQL query it compiles and
                      executes no matter weather there is a change in
                      the query syntax or not. for eg if you are inserting
                      1000 employees your insert query will remain same but
                      Statement object will compile your insert query again
                      and again for 1000 times and runs.
java.sql.PreparedStatement -> this type of statement is designed in such a way
                      that it compiles only when there is a syntatical change
                      in your query. for the same eg above mentioned this will
                      compile the insert statement once and executes it
```

```
                    1000times.
java.sql.CallableStatement -> this is sub interface of PreparedStatement but
                    this is been designed to call up PLSQL stored procedures
                    and functions.


Mysql
driver = com.mysql.jdbc.Driver
url = jdbc:mysql://localhost:3306/java_april2018


Oracle details
drivername=oracle.jdbc.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:xe

oracle downloads

http://www.oracle.com/technetwork/developer-tools/apex/downloads/download-
085147.html

try this as well

http://www.oracle.com/technetwork/developer-tools/apex/application-express/apex-
archive-42-1885734.html


apache tomcat download
https://tomcat.apache.org/download-80.cgi
download the .zip file from above link and extract it and use it

hibernate downloads
http://hibernate.org/orm/downloads/

spring downloads
http://maven.springframework.org/release/org/springframework/spring/

mysqlworkbench downloads
step1 download and install mysql server from below link
https://dev.mysql.com/downloads/mysql/
step 2 download workbench
https://dev.mysql.com/downloads/workbench/
please install prerequistes for mysql and then install the workbench
go through a video in youtube to install it
```

https://www.youtube.com/results?search_query=install+mysql+workbench+on+windows+7

commons logging for spring
http://www.java2s.com/Code/Jar/c/Downloadcommonslogging1211jar.htm

AOP Aspect jars
http://java2s.com/Code/Jar/a/Downloadaspectj169jar.htm
http://www.java2s.com/Code/Jar/a/Downloadaspectjweaverjar.htm

REST Jars
https://jersey.java.net/download.html

9 JSP Implicit Objects
---
out - JspWriter
session - HttpSession
application - ServletContext
request - HttpServletRequest
page - Object
response - HttpServeltResponse
config - JspConfig
exception - Throwable
pageContext - PageContext