1.String / String buffer.

***2.Threads***

***3.object creation***

***4.orderof execution of members***

**5.Class loaders.**

**6.1.5 features , 1.6 features.**

**7.Annotation**

**8.Serialization**

**9.methods in object class.**

**10. Collections.**

**11.OOPs.**

**12.Marker interface.**

**13.Inner classes.**

***1.String and Stringbuffer:***

**1.what is the Differrence between String and stringBuffer?**

**Ans**: 1. String is immutable where as string buffer is mutable.

2.String is asynchronous.stringbuffer is Synchroniged.

3.Both are thread safe.

**2.what is mutable and immutable?eventhough string immutable it’s look like mutable?**

**Ans:**

* Mutable means we can modify the data on objects.
* Immutable means we can’t change the data on objects .

**3.How can you create immutable objects?**

**Ans:**

* Take private properties
* Take only public getter methods.
* Take parameteriged constructor.
* Make class as final.

--By using setter method,we can modify the data on object.

--If we remove setter method,other person doesn’t modify the data on object.

--we have to use constructors.

Final class Data{

**private** String name;

**private** String age;

**private** String qual;

**public** Data(String name,Stringage,Stringqual)

{

**this**.name=name;

**this**.age=age;

**this**.qual=qual;

}

**public** String getName() {

**return**name;

}

Write Getter methods

}

**publicclass**ImutableTest {

**publicstaticvoid** main(String[] args) {

Data data =**new** Data("ABC","23","MA");

System.*out*.println(data.getAge());

Data data2 = data.setAge("43");

System.*out*.println(data2.getAge());

}

}

**4. How can you create immutable objects like String?**

*(if it try to modify then it creates new object)*

**Ans**:Provide setter method that creates new object with new content.

Public Data setAge(String age){

Data data=new Data (this.name,this.age,this.qual);

Return data;

}

**5.what is Asynchronous and synchronized?**

**Ans:**

**Asynchronous** : The object can accessed by multiple threads at single time is called asynchronous.

**Synchronized :** The object can accessed by one thread at a time.

**6.How to Synchronized your java class?**

**Ans:**Two ways 1.synchroniged block

2.Synchroniged keyword.

**7.what is Synchronization? How can you create synchronization in java?what is object level lock and what is class level lock?**

**Ans:**

Thread T1=new thread();

Thread T2=new thread();

--T1 accessing method1-synchroniged method1().

--T2 accessing normal method and class level synchronized method But method level synchronized methods can’t access because t1 has lock on method1(t1 has object lev ellock).

Object level lock example:synchronized method1().

Class level lock example:synchronized static method2().

--T2 can access class level synchronized methods.

Thread T3=new thread();

--T3 can’t access class level synchronized methods becoz t2 has lock on method2.

***Class level lock:****If a thread wants to execute a static synchronized method then it required class level lock.*

*--while thread executing a static synchronized method then the remaining threads are not allowed to execute any static synchronized method of that class simultaneously.but remaining threads are allowed to executes*

**8.What is Thread safe?**

**Ans:**

* One thread should not effect another thread (operations of one thread should not effect on operations of another thread) is called thread safe.
* All local variables are always thread safe.
* In servlet,instance variables are not thread safe.
* Static variables also not thread safe

**9.How can you make servlet thread safe?**

**Ans:**By default servlet is not thread safe becoz of instance & static variables.

* Use only local variables .
* If you want to use instance variables ,use them but not writable.
* If you want to modify instance variables then use single thread model.
* We can use synchronization for servlet to make it thread safe.

These are the ways using make servlet as a thread safe

**Note:**If you want to make static variables as thread safe use synchronization.

**10.What is synchronization ?In how many ways you can implement ?which is better way?**

**Ans:**

Allowing one thread at time is synchronization.

Two ways we can implement 1.Synchronization method

2.Synchronization block.

* To execute any statements that will lock.

**Synchronized(this){**

//which statements we want to make synchronize that statements we have to write.

**}**

* **This**  will say these are current class objects statements .No other object can’t access.

**Synchronized(new service()){**

**//**statements

**}**

**New service():** It will create another object .It will lock another object not current object.

* Here “**this”** helpfull for providing lock to the thread for accessing synchronized statements.
* Synchronized is the modifier apllicable only for ***methods &blocks***.we can’t apply for classes & variables.
* The Main advantage of synchronized keyword is we can resolve data inconsistency problem.
* Disadvantage is it increases the waiting time of the threads &effects performance of the system. Hence there is no specific requirement it’s never recommended to use synchronized keyword.

**8) From where all exception methods are displayed when you call printstackTrace()?**

**Ans:** Stack which is associated with current thread .

--Thread(doGet(),doPost(), service(),dao()) :**printStackTrace()** will print all stack information all methods execution.

**Q) Conversions**

***String to Int Conversion*:-**

int I = integer.valueOf(“24”).intValue();

int x = integer.parseInt(“433”);

float f = float.valueOf(23.9).floatValue();

**Int to String Conversion:-**

String arg = String.valueOf (10);

**9)How to declare synchronized block to get class level lock?**

**Ans:**

Synchronized(classname.class){

//statements

}

**10).what is the advantage of synchronized block over synchronized method?**

**Ans:**It reduces the waiting time of the threads & improves performance of the system.

**10.What is the purpose of the toString () method in java?**

Ans: ->we can use this method to find string representation of an object.

->whenever we are typing to print any object reference internally toString() will be executed.

**11.What is the use of intern () method of String in java?**

**Ans:**By using heap object reference if you want to get corresponding scp object reference then we should go for intern().

ex:String s1=new String("nnr")

String s2 =s1.intern();

sop(s1==s2);false

String s3="nnr";

sop(s3==s2);true

**12.what are advantage of String constant pool?**

**Ans**:

Instead of creating a seperate object for every requirement we can create only one object in scp and we can reuse the same object for every requirement.So that performance and memory utilization will be increased.

**13. What is the me aning of immutable in terms of String?**

**Ans**: Once we created a string object we can't perform any changes in the existing object.

If we are trying to perform any changes with those changes a new object will be created.this behaviour is nothing but immutable of String Object.

ex:String s=new String("NNR");

s.concat("chowdary");

system.out.println(s);



**14.Why String objects are immutable in java?**

**Ans**:In this case of String several references can pointing to the same object.By using one reference,if we are performing any change in the existing object the remaining references will be impacted.to resolve this problem sun people declared as string objects are immutable.According to this once we created a string object we can't perform any changes in the existing object.

**15.How many ways we can create a sting object?**

**Ans:**

**Two ways** 1.by using new operator

ex:String s=new String();

2.by using string literal.

String s="NNR" ;

**16.How many Objects will be created in the following code?**

**String s1=”Cybage”**

**String s2=”Cybage”**

**String s3=”Cybage”**

Ans:one object

**17.Why java uses the concept of String literal?**

ans:

**18.How many Objects will be created in the following code?**

String s1=new String (”Cybage”);

**Ans**: in this case two objects will be created

1.One object is created in heap memory.

2.Another object is created in string constant pool..

s1 is always pointing to heap object.

heap scp

s1=cybase cybase

**19.what is the difference b/w following?**

**String s=new String("NNR"); String s="NNR";**

--In this case two objects will be --**In this** case only one object will be

created one is in heap and other is created in scp and 's' is always pointingin scp and 's' is always pointing to to that object.

heap object.

heap scp scp heap



**Note : G.c** is not allowed to access in scp area hence eventhough object doesn't have any reference variable still it is not eligible for G.c,if it is present in scp.

-All objects present on scp will be destroyed automatically at the time of JVM shutdown.

-There is no chance of two objects with the same content in scp i.e, duplicates objects are not allowed.

-for every string constant compulsory one object will be created in scp.

-bocoz of some runtime operation if an object is required to created that object should be created only on heap but not in scp.

***2.Threads:***

* We can define a thread in the 2 ways

1.by extending thread class2.By implementing Runnable interface

**Q) Thread Class**

Methods: -

|  |  |
| --- | --- |
| getName() | run() |
| getPriority() | Sleep() |
| isAlive() | Start() |
| join() |  |

**Q) Object class**

All other classes are sub classes of object class; Object class is a super class of all other class.

Methods: -

|  |  |
| --- | --- |
| void notify() | void notifyAll() |
| Object Clone() | Sting toString() |
| Boolean equals(Object object) | Void wait() |
| void finalize() | void wait(long milliseconds, int nanoseconds) |

* **I**f we write wait() and notify(),we have to keep the methods in synchronization(block or method).
* InterruptedException is compiletime exception.
* **IllegalMonitorException** raised Why becoz wait() and notify() methods are not inside synchronized.Thats why we have to make synchronization.
* If we call yield.it is going to be runnable state.
* Current thread is going to be wait untill t2 is completed.(t2.join()).
* Join(),sleep(),yield() available in the Thread class.
* Join() is not static method.
* Sleep() & yield() are static methods.
* If we **override start()** then start() will be executed just like a normal method call & no new thread will be created.
* ***Overloading of run()*** is possible,but thread class start() will always call no argument run() only.the other run() we have to call explicitly like a normal method call.

**9.WHAT is the differrence between t.start() & t.run()?**

**Ans:**

* A new thread will be created by t.start().That thread is resposible to execute run().
* But in case of t.run() no thread will be created & run() will be executed just like a normal method call.
* Impartence of thread class start() is

Class thread

{

Start()

{

1.Register this thread with thread sceduler & perform other initialization activities.

2.run().

}}

* If we are not overriding run() method.then thread class run() will be executed which has emty implementation & hence we won’t get any output.

**Note:***Highly recommended to override run() to define our job.*

**9. what is producer and cousumer problem in threads. how can you**

**implement it ?**

**Ans:**

**10.Explain about Thread Priority?**

**Ans:**

* Every thread in java has some priority but the range of thread priorities is “1 to 10”.
* Thread class defines the following constants to define some standard priorities
* THREAD.MIN.PRIORITY 1
* THREAD.NORM.PRIORITY 5
* THREAD.MAX.PRIORITY 10
* Thread scheduler will use these pririorities while allocating cpu
* If two threads having same priority then we can’t expect exact execution order,it depends on thread scheduler.
* Default priority for main thread is 5.But for all the remaining threads it will be inheriting from the parent.
* To get & set priority of thread.

Public final int getPriority().

Public final void setPriority(int p).

**11.what are the yield, join, sleep methods?**

**Ans:**

These methods to prevent the execution.

1**.yield():**

* yield method causes,To pause current executing thread for giving the chance to remaining waiting threads of same priority.
* If there are no waiting threads or all waiting threads have low priority then the same thread will continue it’s execution once again.-+

**2.join():**If thread t1 executes t2.join() then t1 thread will entered into wiating state untill t2 completes.then t1 will continue its execution

**3.sleep():**If a thread don’t want to perform any operation for a perticular amount of time.when we use this we should handle interrupted exception other wise we will get compile time error.

**10. how can you produce deadlock using two threads.**

**11. What are the different exceptions related to threads.**

**12. what are the methods available in Thread class.**

**Ans:**

Methods: -

|  |  |
| --- | --- |
| getName() | run() |
| getPriority() | Sleep() |
| isAlive() | Start() |
| join() |  |

**13. what are the methods available in Object class related to threads?**

**why wait,notify,notifyall methods are available in Object class.**

**Ans:**

* void notify() **2**. void notifyAll() **3.** Void wait()
* wait(),notify(),notifyall() available in object class.but not thread class. Becoz Threads are required to call these methods on any shared object (java.lang.object).

**14. when IllegalMonitorStateException is going to occur ?**

**Ans:**

IllegalMonitorException raised Why becoz wait() and notify() methods are not inside synchronized.Thats why we have to make synchronization.

**15. differences b/w sleep and wait**?

**Ans:**

* Sleep() is going to wait untill sometime.sleep(10000) is defenatly wait untill 5 seconds.
* Wait() is going to wait untill call notify().wait(30000) is not wait 30sec.it is defenatily wait when other person call notify() immediately come out.

**16. wait method releases lock or not ? when thread is going to wait state?**

**17. Thread life cycle.**



* *Once we created a thread object then it is said to be in new state or born state.*
* *If we call start() method then the thread will be entered into ready or runnable state.*
* *If Threadscheduler allocates cpu,then the thread will entered intorunning state.*
* *If run() method completes then the thread will entered into deadstate.*

**18.how can you create deadlock?**

**Ans:**If two threads are waiting for each other forever.Such type of situation is called “Deadlock”.

In case of deadlock waiting threads never end.

**How to kill thread?**

**Ans:**stop().

**19.how to create Deamon thread?**

**Ans:**

* The Threads which are executing in the background are called Daemon treads”

Ex:Garbagecollector

* The main objective of Daemon threads is to provide support for non-daemon threads.
* t1.setDaemon(true).
* t1.start().

Output:created daemon thread

* T1.start()
* T1.setDaemon(true)

Output:IllegalTreadstateException

--We have to make Daemon after creating thread

--don’t do after starting thread.

**package** com.slokam.corejava;

**publicclass** ThreadTest {

**publicstaticvoid** main(String[] args) {

Data data = **new** Data();

Producer producer = **new** Producer(data);

Consumer consumder = **new** Consumer(data);

Thread t1 = **new** Thread(producer);

Thread t2 = **new** Thread(consumder);

t1.setDaemon(**true**);

t1.getName();

producer.setT(t2);

t1.start();

t2.start();

}

}

**class** Data {

**privateint**data;

**privateboolean**avail=**false**;

**synchronizedpublicvoid**insert(**int** data)

{

**if**(avail==**true**)

{

**try** {

wait(30000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**this**.data=data;

avail=**true**;

notifyAll();

}

**synchronizedpublicint** using()

{

**if**(avail==**false**)

{

**try** {

wait();

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

avail=**false**;

notifyAll();

**returnthis**.data;

}

}

**class** Producer **implements** Runnable{

**private** Data data ;

**private** Thread t2;

**publicvoid** setT(Thread t) {

**this**.t2 = t;

}

**public** Producer(Data data)

{

**this**.data=data;

}

@Override

**publicvoid** run() {

**for**(**int** i=1;i<=20;i++)

{

data.insert(i);

**try** {

Thread.*sleep*(5000);

//t2.join();

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

**class** Consumer **implements** Runnable{

Data data ;

**public** Consumer(Data data)

{

**this**.data=data;

}

@Override

**publicvoid** run() {

**for**(**int** i=1;i<=20;i++)

{

System.*out*.println(data.using());

}

}

}

**3.Object Creation:**

**1)How many ways are there to create an object?**

i) Using **‘new’ operator:**

Test s=new Test();

ii) **Factory method**:

Thread t=Thread.currentThread();

iii) **newInstance()**:

Class c=Class.forName(“Test”);

Object obj=c.newInstance(); creates Test class Object.

Test t=(Test)obj;

iv) **clone():**

Test t1=new Test(10,20);

Test t2=t1.clone();

v) **Deserialization**:

FileInputStream fis=new FileInputStream(“Test.txt”);

ObjectInputStream ois=new ObjectInputStream(fis);

UserDefSerCls uds=new UserDefSerCls(,” ”,);

Ois.readObject(uds);

Ois.close();

1. **How many ways to create java objects**

A. a. new operator.

b. newinstance().

c. clonning.

d. serialization / deserialization

A. newinstance() method is available in class called Class.to create

object for Class we have multiple ways.

a. [ClassName].class ex: Class cla = ThreadTest.class;

b. [object].getClass(); ex: Class cla = t2.getClass();

c. using Class.forname() method.

example for creating object by using newinstance();

cls.newInstance(); throws InstantiationException ,

IllegalAccessException

**what is cloning ?**

**what are the types of cloning ?**

**pls write code for implementing shallow cloning and deep cloning .**

**Ans:**

**what is the importance of cloning ?**

Note::For primitive values java supports pass by value.

for objects java supports pass by reference.

**4.Order of execution of java members**

**members of a class**

1. properties(static and instance)

2. methods (static and instance)

3. contructors.

4. blocks. (static and instance)

//first variable and property.

**order**:

1. static properties are initialized.

2. static blocks are executed.

3. static methods are executed only when we call them.

we can use static methods to initialize properties.And we call

static methods with in the static block.

**Note**:: static methods are called before blocks if we use them to

initialize the properties.

4. instance properties are initialized .

5. instance blocks are executed.

6. constructors are executed.

7. instance methods are executed only when we call by using instance.

**Note**:: instance methods are called before constructor and before

blocks if we use them to initialize instance properties.

we can not call instance members in static members why bcz by the

time we are executing static members there is no guarenty of object

availability. Revers is possible.

**5.Class Loaders::**

**1 what is class loading ?**

**loding** .class files into jvm is nothing but class loading.

Class loaders are responsible for loading .class files available

in hard disk into JVM.

**2 Class loaders are three types::**

1. Bootstrap loader.

2. Extension loader.

3. Class path or System loader.

* Bootstrap loader loads all .class files related to JAVA.
* Extension loader loads all .class files available in ext folders.
* Classpath loader loads all .class files available in class path.
* Bootstrap loader can not see the class related to ext and claapath

loaders.

* extension loader can see the classes related to ext and bootstrap.
* classpath loader can see the classes related to classpath,ext,bootstrap loaders.
* Dynamic class loading is possible by using **Class.forname** method.

//Bring .class files into ram.(class loading)

//Bring java s/w into ram.(bootstrap)

//bootstrap load is load the jvm into ram.

//Bring java related files into ram.

--How can u change peram size?

--Before load apache how much memory require into ram?

**3 is possible to create object for class before static block ?**

**Ans:**Static method assign into static property.In that static method we can create object.

Public class TestExecution{

Private static int teststatic=testStatic();

Static{

s.o.p();

}

Public static int testStatic(){

s.o.p();

return 10;

}

Public static void main(){

s.o.p();

}

}

**--**

**{**

**//instance block**

**}**

Private string abc=getData();

Public string getData(){

Return “nnr”;

}

* Static methods are not going to execute automatically.
* Before static block we can execute static methods where intialize static variables we can call static methods to initialize static variable.

**61.5 Features.**

1. Generics.//it is type safety.when we working with multiple objects we can see error.thats why we will use generic.

2. static imports

3. varargs[]

4. Enums //we can make constants better way by using enum

5. annotations.

6. Enhanced for loop.

7. Auto boxing.

8. concurent programming.

// With out main method print value.

// Execution flow missed point

// Notes for annotations.

// Queue example program.

// Real time questions.

**Note** :: In class static property and static block executed which ever appears first.

In instatce members also executed first which ever appears first.

(not including constructor)

**7.Annotations:**

***Steps to create Annotation classes::***

1. Creating annotation::

a. Public @interface TestAnnotation [Name of the annotation ]

b. use Target annotation.

c. use Retention policy.

2. Using annoations::

@TsestAnnotation in other classes .we can apply

this annotation to properties,methods,classes,constructors,

parameters depends upon the Target we provide at the time of annotation creation.

3. Processing the annotations::

a. Get target object to verify whether our annotation has applied.

b. Get all methods availble in the target object.

Method[] methods = apply.getClass().getMethods();

c. verify each method that got applied with our annotation.

TestAnnotation anno = method.getAnnotation(TestAnnotation.class);

d. If applied call the method.

method.invoke(apply);

**1.How can you create annotation?**

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Retention(RetentionPolicy.*RUNTIME*)

@Target(value={ElementType.*METHOD*, ElementType.*CONSTRUCTOR*})

**public@interface**Annotation {

}

**publicclass** AnnotationApply {

**private** String str1;

**private** String str2;

@Annotation

**public** String getStr1() {

System.*out*.println("getStr1");

**return**str1;

}

**publicvoid** setStr1(String str1) {

**this**.str1 = str1;

}

@Annotation

**public** String getStr2() {

System.*out*.println("getStr2");

**return**str2;

}

**publicvoid** setStr2(String str2) {

**this**.str2 = str2;

}

}

**package** com.slokam.corejava.annotation;

**import** java.lang.reflect.InvocationTargetException;

**import** java.lang.reflect.Method;

**publicclass** AnnotationProcessor {

**publicstaticvoid** main(String[] args) {

AnnotationApply apply = **new** AnnotationApply();

Method[] methods = apply.getClass().getMethods();

**for** (Method method : methods) {

Annotation anno = method.getAnnotation(Annotation.**class**);

**if**( anno != **null** )

{

**try** {

method.invoke(apply);

} **catch** (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (InvocationTargetException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} } } }

**2.When you are using retention package?**

**3.what is annotation?**

**8.Serialization:**

**A. what is serialization ? pls write code for serializing object.**

**Ans:**Serialization is a process of converting object state into bytestreem.

**B. what happens if you dont take implements serializable?**

**Ans:** Notserializable Exception arrises.

**C. what is the alternative for serializable?**

**Ans:**Externalizable interface

There are two way to implement serializatoin.

1. using serializable interface.

2. using externalizable interface.

we can provide *custamization* while serialization using *externalizable*. this interface have two methods.

1. readexternal

2. writeexternal

* readexternal method is called before deserialization.
* writeexternal method is called before serialization.

**D. what kind of properties are not serialized?**

**Ans:** native , static , transient variable are not serialized.

**E. what is serialversionUID?**

**Ans:**

* jvm provides serialversionUID by default for every class. value

for this variable is changed by jvm every time we change the clasos.

* When serialization process is happening along with the values version

id is also serialized.

* When desrialization process is happening then jvm check for the version

id of serialized object and versionid of current class , If both are

same then it deserialize properly.If not it rises invalidclass exception.

* To avoid this situation you can take your serialeversionUID in your

class.then jvmdoesnot maintain default serialversionUID.So for small

changes we are not going to change the versionid, so we dont get any

invalidclassexception.

**Note**: if you try to serialze one , all internal objects are serialize.

if those objects are not imlementedserializable then it rises runtime

excption "NotSerializableException".

F**.write code for serialization?**

package com.slokam.serialization;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.Serializable;

public class Account implements Serializable{

private Person person = new Person("JLKSDFLK");

public static final long serialVersionUID = 43;

private String acNo;

private final String address="qwerqr";

private String name;0

private String lastName;

public Account()

{

}

public Account(String acNo,String address,String branch,String name)

{

this.acNo=acNo;

//this.address=address;

//this.branch=branch;

this.name=name;

}

public String getAcNo() {

return acNo;

}

public String getAddress() {

return address;

}

public String getBranch() {

return branch;

}

public String getName() {

return name;

}

}

**package** com.slokam.serialization;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import**java.io.ObjectOutput;

**import** java.io.ObjectOutputStream;

**publicclass** TestSerialization {

**publicstaticvoid** main(String[] args) {

Account acct = **new** Account("24234", "adfasd", "branch", "name");

**try** {

FileOutputStream fos = **new** FileOutputStream("D:/abc.ser");

ObjectOutputStream oos = **new** ObjectOutputStream(fos);

oos.writeObject(acct);

oos.close();

fos.close();

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**G write code for deserializable?**

package com.slokam.serialization;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.ObjectInputStream;

public class Deserialization {

public static void main(String[] args) {

try {

FileInputStream fis = new FileInputStream("f:\\abc.ser");

ObjectInputStream ois = new ObjectInputStream(fis);

Account acct =(Account) ois.readObject();

System.out.println(acct.getAcNo());

System.out.println(acct.getAddress());

//System.out.println(acct.getBranch());

System.out.println(acct.getName());

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

* When we prefered customization then we should go for externalization.
* When we don’t prefered customization then we should go sor serialization.
* We can serializable data of object.
* We can’t serializable data of class.\\static not participate in serialization
* Native objects are not serializable.the objects are created by os that’s it is native object .
* Native objects are fis,collections.
* If don’t take serialversion Id,jvm internally take id,if any change every time jvm will take new id.
* Version Id protect our changes
* I want to send data one location to another location serialization required.
* To send one system to another system required serialization.

**9.Collections:**

**Arrays:**

* Array is an object,that object contains space to store multiple values.

***Declaration of array***

* Int a[]=new int[5]
* Int b[]={22,55,66};
* Int[] c=new int[7];
* Int[] d={11,22,33};
* Arrays can be multiple dimentional.
* **Ex:**int data[][]=new int[2][3];
* Arrays can not grow at runtime, size can be declared at initialization. you can provide only similar data type .
* Arrays supports only homogeneous data elements.

To overcome above problems introduced collections

**Collections:**

* Collections are growable array
* All collections uses internally uses object array(object[])
* Collections can hold both Homogeneous & Heterogeneous objects.

**Collection vs Collections:**

* Collection is an interface,can be used to represent a group of individual object as a single entity
* Collections is an utility class present in java.util package to define several utility methods for collections

**Collection Framework:**

* It defines several classes & interfaces ,which can be used to represent a group of objects as a single entity.

**List:**

* If we want to represent a group of individual objects where insertion order is preserved & duplicates are allowed.Then we should go for List
* We can differentiate duplicate objects by using index.

**ArrayList:**

* The speciality of array List is very fast where we are retrieving data.
* If we want delete, insert multiple times it is very slow. Becoz it require several shift operations
* Array List is Asynchronous that is the reason multiple threads can access your array list.
* ArrayList & vector classes implements Random Access interface, so that any random element we can access with same speed. Hence, if our frequent operation is retrieval operation then best suitable data structure is Array list

**Vector:**

* Vector has all qualities of arraylist but it is synchronized ,that is the reason multiple threads can not access at given point of time.
* Insertion order is preserved
* Duplicate objects are allowed
* Heterogeneous objects are allowed

**Linked List:**

* Linked List is Asynchronous
* Linked List is faster than array List in case of deleting & inserting.
* It is slower than array List in case of retrieving object
* Null insertion is possible
* Insertion order is preserved
* Duplicate objects are allowed
* Heterogeneous objects are allowed

**Set List**

1.It has unique nature -Don’t have unique nature

2.insertion order is not preserved -List has insertion order

**3.**duplicate objects are not allowed -allowed

4.It doesn’t contain any method only we have -It has methods

To use collection interface method

**Arraylist Vector**

* No method synchronized -every method synchronized
* Mutliple threads can access arraylist -at any point only one thread is allowed

Simultaneously

* Threads not required to wait & performance - It increasing waiting time of threads

Is high .1.2 & hence performance low.1.0.object

is thread safe

**Set:**

* If we want to represent a group of objects where duplicates are not allowed & insertion order is not preserved. then we should go for set
* Set interface doesn’t contain any method we have to use only collection interface method.

**Hash Set:**

* Hash Set maintains unique key object
* HashSet doesn’t have any order
* To get data from HashSet by using iterator,enumeration
* The underlying data structure is hash table
* Duplicate objects are not allowed
* Heterogeneous objects are allowed
* Null insertion is possible(only once)becoz duplicates are not allowed.
* If we are trying to add duplicate objects ,we won’t to get any c.e or r.error add() simply returns false.
* Insertion order is not preserved & all objects are inserted according to hashcode of the objects.

**TreeSet:**

* TreeSet same as Linked List but it maintains sorting order
* It eleminates duplicateion
* Insert order is not preserved .becoz objects will be inserted according to some sorting order
* Heterogeneous objects are not allowed. otherwise we will get “classCastException”

& null insertion is not possible

***Null acceptance:***

* For the non-empty treeset, if we are trying to insert null we will get nullpointerException
* For the empty treeset add the first element null insertion is always possible.
* But after inserting that null,iff we are trying to insert any other,we will get nullpointerexception.
* If we are depending on default sorting order compulsory objects should be homogeneous & comparable otherwise we will get classcastexception
* An object said to be comparable iff the corresponding class implements comparable interface.
* String class & all wrapper classes already implements comparable interface where as Stringbuffer doesn’t implement comparable interface .

**publicclass** Treeset1 {

**publicstaticvoid** main(String[] args) {

TreeSet t=**new**TreeSet();

/\* t.add(new StringBuffer("d"));

t.add(new StringBuffer("a"));

t.add(new StringBuffer("g"));\*/

/\* t.add(new String("f"));

t.add(new String("r"));

t.add(new String("a"));\*/

t.add(**new** Integer(2));

System.*out*.println(t);

}

}

* When we are depending on default natural sorting order internally jvm calls compareTo()
* We can define our own customized sorting order by using comparator

Comparable ment for default natural sorting order

Comparator ment for customized soring order

**Linked Hash Set:**

* LHSmaintains insertion order.
* It is exactly same as hash set except the following differrences

**Hash Set Linked Hash Set**

* Insertion order is not preserved preserved
* 1.2 1.4
* The underlying data structure is hashtable combination of hashtable & LinkedList.

**Hash Set Tree Set**

* It don’t have any order It has sorting order
* Equals() & hashcode() compareTo() & compare()
* Tree Map uses Tree set to maintain key values

**Note:** In every collection class toString() is overridden to return its content directly in the following formate[obj1,obj2,obj3….]

**MAP:**

* Map is interface.It maintains key value pairs.It contains mutliple implementation class.
* Duplicate keys are not allowed,But values can be duplicated
* If we want to represent a group of objects as key-value pairs then we should go for map. both key & value are objects.
* Each key value pair is called Entry.
* There is no relationship b/w collection & map
* Mehtods:

1.Object put(Object key,Object value)

2.Object get(Object key)\\returns value associated with key.not key there returns null

3.void putAll(Map m)

**HashMap:**

* HashMap doesn’t maintain any order
* Null key is allowed (only once)
* Null values are allowed (any no.of times)
* Duplicates not allowed but the values can be duplicated
* Heterogeneous objects are allowed for both keys & values.

**TreeMap:**

Tree Map has sorting order for keys

* Insertion order is not preserved & all entries are inserted according to some sorting order of keys.
* If we are depending on default sorting order compulsory the keys should be homogeneous & comparable otherwise we will get class cast exception.
* If we are defining our own sorting order by comparator then the keys need not be homogeneous & comparable.
* Duplicates keys are not allowed but values can be duplicated

**HashTable:**

* The underlying data structure is hash Table
* Heterogeneous objects are allowed for both keys & values
* Insertion order is not preserved & it is based on hashcode of the keys
* Null is not allowed for both key & values otherwise we will get Null pointer exception
* Duplicates keys are not allowed but values can be duplicated

**LinkedHashMap:**

It has insertion order



**HashTable HashMap**

* Doesn’t contain null we can take key as well as value is null
* Synchronized Not synchronized.
* No orders No order
* To maintain userdefind keys in HashMap ,we have to override equals() & hashcode()

**HashMap LinkedHashMap**

* The underlying d.s is hash table combination of hash table & Linked List.
* Insertion order is not preserved insertion order is preserved
* 1.2 1.4

**Note**: All three classes are Asynchronous. There is synchronized map implementation is Hash Table.

**1.Without knowing key value , How to print map date?**

**Ans**: public class MapTest{

Public static void main(){

HashMap map=new HashMap()

Map.put(“one”,”hyderabad”);

Map.put(“two”,”bombai”)

Map.put(“three”,”bangalore”)

s.o.p(map.get(“three”);

set set=map.keySet();

s.o.p(set);

iterator itr=set.iterator();

while(itr.hasNext()){

s.o.p(itr.next())

s.o.p(map.get(itr.next()));

}}**}**

**2.How to know objects are implements or extend?**

**Ans**:instanceOf

A implements B

B implements C

A a=new A();

(a instanceOf B)\\true

(a instanceOf c)\\false

Public interface validate{

}

Public class Acount Implements validate{

//statements

}

Public class TangedInterfaceProcess{

P s v main(){

Acount act=new Acount();

If(act instanceof validate)

{

Act.getAcno();

//logic processing

}

}

}

**3.what is the importance of equals() & hashcode()?**

**4.What are the returns type for equals() & hashcode()?**Equals and HashCode methods in Java are two fundamental methods from java.lang.Object class, which is used to compare equality of objects, primarily inside hash based collections such as Hashtable and HashMap. Both equals() and hashCode() are defined in java.lang.Object class and there default implementation is based upon Object information e.g. default equals() method return true, if two objects are exactly same i.e. they are pointing to same memory address, while default implementation of hashcode method return int and implemented as native method. Similar default implementation of toString() method, returns type of class, followed by memory address in hex String.

**5.what is differrence between comparator and comparable?**

**Ans:**

**Comparable comparator**

**1**.we can use comparable to define default 1 we can use comparator to define

natural sorting order1 customize sorting oreder

2.java.lang.package 2.java.util.package

3.Defines only one method 3.Define two methods

compareTo() 1.compare() 2.equals()

**Differences between the comparator and the comparable interfaces :**  
  
***1)*** In comparable ,Only one sort sequence can be created while in comparator many sort sequences can be    created .  
***2)*** Comparator interface in Java has method public int compare (Object o1, Object o2) which returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second. While Comparable interface has method public int compareTo(Object o) which returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.  
***3)*** If you see then logical difference between these two is Comparator in Java compare two objects provided to it , while Comparable interface compares "this" reference with the object specified. .  
***4)*** One has to modify the class whose instances you want to sort while in comparator one build a class separate from the class whose instances one want to sort .  
***5)*** Comparator in Java is defined in java.util package while Comparable interface in Java is defined in java.lang package, which very much says that Comparator should be used as an utility to sort objects which Comparable should be provided by default.  
***6)*** Comparable in Java is used to implement natural ordering of object. In Java API String, Date and wrapper classes implements Comparable interface.Its always good practice to override compareTo() for value objects.  
***7)*** If any class implement Comparable interface in Java then collection of that object either list or Array can be sorted automatically by using Collections.sort() or Arrays.sort() method and object will be sorted based on there natural order defined by CompareTo method.  
***8)*** Objects which implement Comparable in Java can be used as keys in a SortedMap like treemap or elements in a SortedSet for example TreeSet, without specifying any Comparator.  
**Situations when to use Comparable & Comparator**  
  
1) If there is a natural or default way of sorting Object already exist during development of Class than use Comparable. This is intuitive and you given the class name people should be able to guess it correctly like Strings are sorted chronically, Employee can be sorted by there Id etc. On the other hand if an Object can be sorted on multiple ways and client is specifying on which parameter sorting should take place than use Comparator interface. for example Employee can again be sorted on name, salary or department and clients needs an API to do that. Comparator implementation can sort out this problem.  
  
2) Some time you write code to sort object of a class for which you are not the original author, or you don't have access to code. In these cases you can not implement Comparable and Comparator is only way to sort those objects.  
  
3) Beware with the fact that How those object will behave if stored in SorteSet or SortedMap like TreeSet and TreeMap If an object doesn't implement Comparable than while putting them into SortedMap, always provided corresponding Comparator which can provide sorting logic.  
  
4) Order of comparison is very important while implementing Comparable or Comparator interface. for example if you are sorting object based upon name than you can compare first name or last name on any order, so decide it judiciously. I have shared more detailed tips on compareTo on my post how to implement CompareTo in Java.  
  
5) Comparator has a distinct advantage of being self descriptive for example if you are writing Comparator to compare two Employees based upon there salary than name that comparator as Salary Comparator, on the other hand compareTo()  
  
*So in Summary* if you want to sort objects based on natural order then use Comparable in Java and if you want to sort on some other attribute of object then use Comparator in Java.

**6.what is hashing technique?**

**Ans:**Hashing is designed to solve the problem of needing to **efficiently** find or store an item in a collection.

**7.How hashset maintains unique nature?**

**7.what is comparision logic?**

Ans:We have to provide ascending,descending order.

**8.How to provide sorting for collection?**

**9.What is compareTo()?**

**Ans:** obje1.compareTo(obj2)

* Returns –ve iff obj1 has to come before obj2
* Returns +ve iff obj1 has to come after obj2
* Returns 0 iff obj1 & obj2 are equal(duplicate)
* *When are depending on default nature sorting order internally jvm calls compareto()*

**10.what is differrence b/w compareTo() & equals()?**

**Property enumeration Iterator ListIterator**

**1.It** is legacy yes No No

2.It is applicable only for legacyclasses for any collection only for listobjects

Objects

3.movement only forward only forward bi-directional

4.how to get it? By uing elements() method BU iterator() read/replace

/remove/add

5.method hasMoreElements() hasNext(),next() 9 methods

Nextelement() remove()

**11.what is the differrence b/w compareTo() & compare()?**

**Ans:**

**10.how Treeset work internally?**

**Ans:**we are going to write pojo class,pojo objects are placed into tree set .tree set calls compare().(By pojo class we can use no.of objects).

Treset contains no.of objects.

**11.what happens one object place into Treeset/Hashset?**

**Ans:**

* To maintain unique nature hashset calls hashcode() equals().
* To maintain unique nature treeset calls compareTo().
* Treeset is very slow becoz every time it will compare all objects.
* When we enter one object into treeset it will make sorting order.every time it will do sorting order.thats why it is very slow

**12.where you use comparator & comparable?**

* Default equals() don’t have capability to compare two objects contents are same or not.
* Every object have same hashcode they will place into same bucket in set.
* Every object has different hashcode they will place into different buckets In set.
* If don’t provide hashcode(),it will search for super class hashcode().It indicates different objects.
* If both objects are same content both objects contains same hashcode.
* If both objects have same hashcode both objects may or may not same.
* If two objects hashcode different the two objects are complete different.
* When object place into hashset,the hashcode() will be called by automatically.
* If don’t implement comparable,It will not find comparision logics(compareTo(Object o){})
* If we provide Generic ,directly we can use Address class into compareTo(Address a) as argument
* If you use Comparator,Treeset will call compare().
* If you use comparable,Treeset will call compareTo().

**what is the hierarchy of collections?**





**HashTable HashMap**

* Doesn’t contain null we can take key as well as value is null
* Synchronized Not synchronized.
* No order No order
* To maintain userdefind keys in HashMap ,we have to override equals() & hashcode()

**15.How to hashMap maintain key unique?**

**Ans:**It use internally Hashset

**16.how to make ArrayList sorting order?**

**Ans:**

To maintain ArrayList sorting order,we have to provide Comparator

* **publicclass** Sorting {

**publicstaticvoid** main(String[] args) {

List<String> list = **new** ArrayList<String>();

list.add("X");

list.add("Z");

list.add("S");

list.add("A");

Collections.sort(list);

//If don’t provide comparator or comparable, shows the error at collections.sort()

s.o.p(list);

}

* **publicclass** Address **implements** Comparator<Address>{

**private** String pincode;

**private** String street;

//**write setters & getters**

@Override

**publicint** compare(Address arg0, Address arg1) {

// **TODO** Auto-generated method stub

**return** arg0.getPincode().compareTo(arg1.getPincode());

}

}

**publicclass** ArrayListsorted {

**publicstaticvoid** main(String args[]){

Address add1=**new** Address();

add1.setPincode("4419999");

add1.setStreet("srnagar");

Address add2=**new** Address();

add2.setPincode("113333");

add2.setStreet("brnagar");

Address add4=**new** Address();

add4.setPincode("21333367");

add4.setStreet("aabrnagar");

Address add3=**new** Address();

add3.setPincode("222444");

add3.setStreet("gaarnagar");

//Phone ph2=new Phone();

List<Address> listadd=**new** ArrayList<Address>();

//Set<Address>listadd=new TreeSet<Address>(new Addresspincodecomparator());

listadd.add(add1);

listadd.add(add2);

listadd.add(add3);

listadd.add(add4);

Collections.*sort*(listadd,**new** Addressstreetcomparator());

**for**(Address address:listadd){

System.*out*.println(address.getPincode()+"::"+address.getStreet());

}

}

}

**Note**:

* For primitive values java supports pass by value.
* For object values java supports pass by refference.

**How can you make collections is immutable?**

**Ans:**

Collections.unModifiableCollection(listAdd);

**How can you make collections is Synchronized?**

**Ans:**

Collections.synchronizedCollection(listAdd);

**How can you sort your ArrayList?**

**Ans:**

By using Collections.sort(listAdd,new AddresspincodeComparator());

**Internal implementation of Set/HashSet (How Set Ensures Uniqueness) : Core Java Collection Interview Question**

Interviewer asked How do you implement Set in  Java . That is , how will make sure each and every element is unique without using Set interfaces or Classes that implements Set Interface .  
  
I gave the answer , although qualified the interview round as well , but the answer is far from satisfactory .  
So I came back to  home and do some research . So finally i got the answer and sharing it with you .  
  
  
**Set Implementation Internally in Java**  
Each and every element in the set is unique .  So that there is no duplicate element in set .  
  
So in java if we want to add elements in the set then we write code like this  
  
public class JavaHungry {  
  
 public static void main(String[] args)  
 {  
 HashSet<Object> hashset = new HashSet<Object>();  
           hashset.add(3);  
 hashset.add("Java Hungry");  
 hashset.add("Blogspot");  
 System.out.println("Set is "+hashset);   
 }  
 }  
*It will print the result* :       Set is [3, Java Hungry, Blogspot]  
Now let add duplicate element in the above code  
  
public class JavaHungry {  
  
 public static void main(String[] args)  
 {  
 // TODO Auto-generated method stub  
  
 HashSet<Object> hashset = new HashSet<Object>();  
          hashset.add(3);  
 hashset.add("Java Hungry");  
 hashset.add("Blogspot");  
**hashset.add(3); *// duplicate elements***  
**hashset.add("Java Hungry");   *// duplicate elements***  
 System.out.println("Set is "+hashset);   
 }  
 }  
  
  
*It will print the result* :       Set is [3, Java Hungry, Blogspot]  
Now , what happens internally when you pass duplicate elements in the  add() method of the Set object , It will return false and do not add to the HashSet , as the element is already present .So far so good .  
  
But the main problem arises that how it returns false . So here is the answer  
  
When you open the HashSet implementation of the add() method in Java Apis that is rt.jar , you will find the following code in it  
public class HashSet<E>  
    extends AbstractSet<E>  
    implements Set<E>, Cloneable, java.io.Serializable  
  
{  
       private transient HashMap<E,Object> map;  
  
      // Dummy value to associate with an Object in the backing Map  
  
      private static final Object PRESENT = new Object();  
  
*public HashSet() {*  
*map = new HashMap<>();*  
*}*  
  
     // SOME CODE ,i.e Other methods in Hash Set  
       public boolean add(E e) {  
            return map.put(e, PRESENT)==null;  
       }  
  
     // SOME CODE ,i.e Other methods in Hash Set  
}  
  
So , we are achieving uniqueness in Set,internally in java  through HashMap . Whenever you create an object of HashSet it will create an object of HashMap as you can see in the italic lines in the above code .  
We already discussed   How HashMap works internally  in java .  
  
As we know in HashMap each key is unique . So what we do in the set is that we pass the argument in the add(Elemene E) that is E as a key in the HashMap . Now we need to associate some value to the key , so what Java apis developer did is to pass the Dummy  value that is ( new Object () ) which is referred by Object reference PRESENT .  
  
So , actually when you are adding a line in HashSet like  hashset.add(3)   what java does internally is that it will put that element E here 3 as a key in the HashMap(created during HashSet object creation) and some dummy value that is Object's object is passed as a value to the key .  
  
Now if you see the code of the HashMap put(Key k,Value V) method , you will find something like this  
  
 public V put(K key, V value) {  
//Some code  
}  
  
The main point to notice in above code is that put (key,value) will return  
  
1.  null , if key is unique and added to the map  
2.  Old Value of the key , if key is duplicate  
  
So , in HashSet add() method ,  we check the return value of map.put(key,value) method with null value   
i.e.  
  
   public boolean add(E e) {  
            return map.put(e, PRESENT)==null;  
       }  
  
So , if map.put(key,value) returns null ,then  
map.put(e, PRESENT)==null      will return true and element is added to the HashSet .  
  
So , if map.put(key,value) returns old value of the key ,then  
map.put(e, PRESENT)==null      will return false and element is  not added to the HashSet .  
  
If you still have any doubts then please write in comments .

**When you are writing equals() method, which other method or methods you need to override?**

Ans:hashcode, is the right answer. Since equals and hashCode has there contract, so overriding one and not other, will break contract between them. By the way this question can lead on interesting discussion, if Interviewer likes to go on deep e.g. he may ask about what are those contracts, what happens if those contracts breaks etc. I like to give an example How equals and hashcode are used in hash based collections e.g. Hashtable, that leaves positive impression more often. You can also mention about compareTo() here to score some additional point, this method should also needs to be consistent with equals, which is another interesting question in our list.

**Hashing :How hash map works in java or How get() method works internally**

Ans:

One of the most darling question of the core java interviewers is How hash map works in java . Most of the candidates rejection chances increases if the candidate do not give the satisfactory explanation . This question shows that candidate has good knowledge of Collection . So this question should be in your to do list before appearing for the interview .  
  
Read also  How Hashset works in java or How it ensures uniqueness in java   
  
HashMap is the key value pair .  To understand hashing we talk about three terms frequently *hashfunction ,hash value  .*  
  
hashCode() function  which returns an integer value is the **Hash function**. The important point to note that ,  this method is present in Object class ( Mother of all class ) .  
  
This is the code for the hash function(also known as hashCode method) in Object Class :  
  
    public native int hashCode();

Here the most important point to note from the above line is that hashCode method return  int value .  
So the **hash value is the int value returned by the hash function .**  
  
The other **important point** to note is that **in Map ,Any class(String etc.) can serve as a key if and only if it overrides the equals() and hashCode() method .**  
    So summarize the terms in the diagram below :  
                  

  
  
  
  
  
  
  
  
  
  
After understanding the terms we are ready to move next step ,

**How hash map works in java** **or How get() works internally in java .**  
  
  
  
**Code inside Java Api (HashMap class internal implementation) for HashMap get(Obejct key) method**   
  
1.                  Public  V get(Object key)  
               {  
2.                   if (key ==null)  
3.                          //Some code  
  
4.                   int hash = hash(key.hashCode());     
  
5.                  // if key found in hash table then return value  
6.                 //    else return null             
               }

**Hash map** **works on the principle of hashing**  
  
HashMap get(Key k) method calls hashCode method on the key object and applies returned hashValue to its own static hash function to find a bucket location(backing array) where keys and values are stored in form of a **nested class called Entry (Map.Entry)** . So you have concluded that from the previous line that **Both key and value is stored in the bucket as a form of  Entry object** . So thinking that Only value is stored  in the bucket is not correct and will not give a good impression on the interviewer .  
  
\* Whenever we call get( Key k )  method on the HashMap object . First it checks that whether key is null or not .  Note that **there can only be one null key in HashMap .**  
 **If key is null , then Null keys always map to hash 0, thus index 0.**  
  
If key is not null then , it will call hashfunction on the key object , see line 4 in above method i.e. key.hashCode()  ,so after key.hashCode() returns hashValue , line 4 looks like  
  
4.                int hash = hash(hashValue)  
  
 , and now ,it applies returned hashValue into its own hashing function . Now this value is used to find the bucket location at which the Entry object is stored . **Entry object stores in the bucket like this (hash,key,value,bucketindex) .**  
  
But the problem arises when two objects have the same hash . Now the role of hashCode() method in the HashMap class ends here . Now the role of equals() method starts .  
  
**The bucket is the linked list effectively . Its not a LinkedList as in a java.util.LinkedList - It's a separate (simpler) implementation just for the map .**  
  
**So we traverse through linked list , comparing keys in each entries using keys.equals() until it return true.  Then the corresponding entry object Value is returned .**  
If you still have any doubts then please write in comments

**10.class & object?**

**class**

**à cl** is a Template that describes the **Kind of State**(The Instance Variables) and **Behavior** (Methods)

**à(Metho** is a blue print of an object .

component means u can use a piece of code like an independent piece.like servlet,EJB...etc

**Object**àbjectpendent piece**.**u can reuse it in any application

**Q) Byte code &JIT compiler:**

Byte code is a highly optimized set of instructions. JVM is an interpreter for byte code. Translating a java program into byte code helps makes it much easier to run a program in a wide variety of environment.

JIT is a part of JVM, it compiles byte code into executable code in real time, will increase the performance of the interpretations.

**Q) Public static void main (String [] args):**

**à*What if the main method is declared as private?***

The program compiles properly but at runtime it will give "Main method not public." Message

***àWhat if the static modifier is removed from the signature of the main method?***

Program compiles. But at runtime throws an error "NoSuchMethodError".

**à**We can write “**static public void**” instead of “**public static void**” but not “**public void static**”.

*à****If I do not provide the String array as the argument to the method?***

Program compiles but throws a runtime error "NoSuchMethodError".

**à Program compiles but throws a runtime error "NoSuchMethodError".ror".hod?t public." M**

It is empty. But not null.

à It is empty. But not null.s a runtime error "NoSuchMe

**Q) Can an application have multiple classes having main method?**

A) Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned. Hence there is not conflict amongst the multiple classes having main method.

**Q) Can I have multiple main methods in the same class?**

A) No the program fails to compile. The compiler says that the main method is already defined in the class.

**11.Exception Handling**

**Runtime Stack Mechanism**

For every thread JVM will create a runtime stack. All the method calls performed by the thread will be

sorted in the corresponding runtime stack. If a method terminates normally the corresponding entry from the

stack will be removed.

After completing all the method calls the stack is empty. Just before terminating the thread JVM will destroy

the corresponding stack.

Ex:

class ExceptionDemo

{

public static void main(String[] args)

{

doStuff();

}

public static void doStuff()

{

doMoreStuff();

}

public static void doMoreStuff()

{

System.out.println("Hi this is Exception ...........Thread");

}

}

**Default Exception Handling**

Ex:

class ExceptionDemo

{

public static void main(String[] args)

{

doStuff();

}

public static void doStuff()

{

doMoreStuff();

}

public static void doMoreStuff()

{

System.out.println(10/0);

}

}

**O/P:-**

When ever an exception raised the method in which it is raised is responsible for the preparation of

exception object by including the following information

Name of Exception.

Description.

Location of Exception.

After preparation of Exception Object, The method handovers the object to the JVM, JVM will check for

Exception handling code in that method if the method doesn’t contain any exception handling code then

JVM terminates that method abnormally and removes corresponding entry from the stack.

JVM will check for exception handling code in the caller and if the caller method also doesn’t contain

exception handling code then JVM terminates that caller method abnormally and removes corresponding

entry from the stack.

This process will be continued until main method and if the main method also doesn’t contain any exception

handling code then JVM terminates main method abnormally.

Just before terminating the program JVM handovers the responsibilities of exception handling to default

exception handler. Default exception handler prints the error in the following format.

Name of Exception : Description

stackTrace

**1.What is Exception?**

**Ans:**

* When unwanted,unexpected event that disturbes normal flow of program is called “Exception”.
* Exception handling doesn’t mean repairing an Exception,we have to define alternative way to continue rest of the program normally this is nothing but Exception Handling.

**Exception**

These are recoverable. Most of the cases exceptions are raised due to program code only.

**Error**

Errors are non-recoverable. Most of the cases errors are due to lack of system resources but not due

to our programs.

**2.what is Rutime stack mechanism?**

**Ans:**

* For every thread JVM will create a runtimestack.
* All method call performed by the thread will be store in the stack.
* Each entry in the stack is called “Activation record” or “stackframe”.
* After completing every method call Jvm deletes the corresponding entry from the stack.
* After completing all method calls,just before terminating the thread jvm destroyeds the stack.

**Exception hierarchy:**

* Throwable acts as a root for entire java exception hierarchy
* It has 2 classes

1.exception

2.Error

**Exception**:Most of the cases exceptions are caused by our program. & recoverable.

**Error:**Most of the cases errors are not caused by user program these are due to lack of system resources.Error are non-recoverable.

**3.what is checked Exceptions?**

**Ans:**

* The exceptions which are checked by compiler for smooth excecution of the program at runtime are called “checked exception”.

Ex:fileNotfoundexception

**4.what is unchecked Exception?**

**Ans:**

* The exceptions are not checked by compiler are called “un-checked exception”.
* Wheather Exception is checked or unchecked it should runtime only.there is no chance of occuring at compiletime.

**5.what is the partially checked vs fully checked?**

**Ans:**

* A checked exception is said to be fully checked iff all it’s child classes also checked.

Ex:**IOException,**

* A checked exception is said to be partially checked iff some of it’s child classes are unchecked.

Ex:**exception,**

**Note:**In java the only partially checked exceptions are:1.Exception 2. Throwable

**Customized Exception handling by Try-catch:**

* We can maintain riskycode with in the try block & corresponding handling code inside catch block.

Try{

Riskycode

}

Catch(\*\* e){

Handling code

**}**

**Note:**with in the try block if any where an exception raised then rest of the try block won’t be executed eventhough we handled that exception.Hence it is recommended to take only risky code with in the try block.

**Various Methods to print exception Information:**

* Throwable class defines the following methods to print exception information.

1.**PrintStackTrace():**this method prints exception information in the following format

Ex: Name of exception:discription

stacktrace

**2.toString():**It prints exceptioninformation in the folowingformat

**Ex:Name of exception:discription**

**3.getMessage():**this method prints only disription of the exception.

**Ex:**discription

**Note**:Default exception handler uses printstackTrace().

Order of catch blocks is from **child to parent** otherwise it gives compiletime error

**Finally block:**

The main purpose of finally-block is to maintain clean-up code which should be executed always.wheather exception raised or not.

* It is never recommended to define clean up code with in the try block becoz there is no gauranty for the execution of every statement
* It is never recommended to define clean up code with in the catch block,bcoz it won’t be executeded if there is no exception

**Return vs finally:**finally block dominates return statement also.Hence If there is any return statement present inside try or catch block.first will be executed & then return statement will be considered

**NOTE:**there is only one situation where the finally block won’t be executedis,when ever jvm shutdown i.e when ever we are using **system.exit(0).**

**Differrence b/w final,finally & finalize:**

***Final:***

* It is a modifier for classes,methods & variables
* If class declared as final,then child class creation is not possible.
* If method declared as final,then overriding of that method is not possible.
* If a variable declared as the final,then reassignment is not allowed becoz it is a constant

***Finally***

* It is block always asociated with try-catch to maintain clean-up code which should be executed always irrespective of wheather exception raised or not raised & wheather handled or not handled.

***Finalize()***

* It is a method which should be executed by garbage collector before destroying any object to perform clean-up code activities.

***NOTE:***when we compare with finalize(),it is highly recommended to use finally block to maintain clean-up code becoz we can’t expect behaviour of the garbage collector

***Throw:***sometimes we can create exception object manually & handover that object to the JVM explicitly by using throw keyword

throw new ArithematicException(“/by zero”);

The main purpose of throw keyword is to hand-over our created exception object manually to the jvm

Class Test class Test{

{ p s v main(){

P s v main(){ throw new AE(“/by zero”);

s.o.p(10/0); }

} }

}

In this case A.E object created internally In this case we created A.E object and

& handover that object automatically by we handover it to the jvm manually

The main(). By using throw key word.

* In generally we can use throw keyword for customized exception
* After throw statement we are not allow to write any statement directly otherwise we will get compiletimeerror saying “unreadable statement”.
* We can use throw keyword only for throwable type otherwise we will get compile time error saying incompatible types

***Throws:***In our program,If there is any chance of raising cheked exceptions compulsary we should handle it,otherwise we will get compiletime error says”unreadable exception” must be caught or declare to be thrown

We can handle this by using the following two ways

* By using try-catch
* By using throws

**By using throws:**we can use throws keyword to delegate the responsibility of exception handling to the caller methods in the case of checked exception.in case of unchecked exception,it is not required to use throws keyword.

**Exception Handling keywords Summary:**

* **Try:** To maintain risky code.
* **Catch:** To maintain handling code
* **Finally:** To maintain clean up code
* **Throw:** To handover our created exception
* **Throws:** To delegate the responsibility

**CustomizedExceptions:**

**Ans:To** meet our programming requirement sometimes we have to create our own exception .such type of exceptions are called “customizedExceptions”.

1 **.Jvm exceptions:**The exceptions which are raised automatically by the jvm when ever a particular event occurs are called jvm Exceptions.

2.**Programmaticexceptions:**The exception which are raised explicitly either by the programmer or by the api developer are called programmatic exceptions

**Exception Propagation:T**he process of delegating the responsibility of exception handling from one method to another method by using throws keyword is called ExceptionPropagation.

**Q) Static block**

Static block which exactly executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.

**Q) Static variable & Static method**

Static variables & methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class.

Static methods can be referenced with the name of the class. It may not access the instance variables of that class, only its static variables. Further it may not invoke instance (non-static) methods of that class unless it provides them with some object.

When a member is declared a static it can be accessed before any object of its class are created.

Instance variables declared as static are essentially global variables.

If you do not specify an initial value to an instance & Static variable a default value will be assigned automatically.

Methods declared as static have some restrictions they can access only static data, they can only call other static data, they cannot refer **this** or **super**.

Static methods cant be overriden to non-static methods.

Static methods is called by the static methods only, an ordinary method can call the static methods, but static methods cannot call ordinary methods.

Static methods are implicitly "final", because overriding is only done based on the type of the objects

They cannot refer “this” are “super” in any way.

**Q) Class variable&Instance variable &Instance methods &class methods**

*Instance variable* variables defined inside a class are called instance variables with multiple instance of class, each instance has a variable stored in separate memory location.

*Class variables*  you want a variable to be common to all classes then we create class variables. To create a class variable put the “static” keyword before the variable name.

*Class methods*  we create class methods to allow us to call a method without creating instance of the class. To declare a class method use the “static” key word.

*Instance methods* we define a method in a class, in order to use that methods we need to first create objects of the class.

**Q) Static methods cannot access instance variables why?**

Static methods can be invoked before the object is created; Instance variables are created only when the new object is created. Since there is no possibility to the static method to access the instance variables. Instance variables are called called as non-static variables.

**12.Inner classes:**

* We can declare a class inside another class,such type of classes are called “inner classes”.
* Inner classes concept introdused in java 1.1 version to fix gui bugs as the part of eventhandling.
* Without existing one type of object if there is nochance of existing another type object then we should go for inner classes concept.

Class Car{

Class wheel

{

}

}

Ex:without existing bank object there is no chance of existing acount object,Hence we have to define acount class inside Bank class

Class Bank

{

Class Acount

{

}

}

**Note:The relationship b/w outer & inner classes is not parent-child relationship.It is has-A relationship.**

*Based on the purpose ?& position of declaration all inner classes are divided into 4 types:*

* Normal or regular inner classes
* Method local inner classes
* Annonymous inner classes(without class name)
* Static nested classes

**Note**:From static nested class we can access only static members of outer class directly.But in normal inner classes we can access both static & non-static members of outer class directly.

1.**Normal or Regular Inner class:**If we declare any named class directly inside a class without static modifier,such type of class is called “normal inner class”

Class outer{

Class inner

{

}

Public static void main(){

s.o.p(“outer class main method”);

}

}

o/p:javac outer.java

o/p:outer class main method

java outer$inner

o/p:Nosuchmethod error:main

**Ex2**:Inside inner classes we can’t declare static memebers hence it is not possible to declare main() & hence we can’t invoke inner class directly from command prompt.

* Inner class can’t have static declarations.

**1.Accessing inner class code from static area of outer class:**

**Or from out side of outer classes:**

**Ans: publicclass** Outer {

**class** Inner{

**publicvoid** m1(){

System.*out*.println("inner classes method nnr");

}

}

**publicstaticvoid** main(String[] args) {

/\*Outer o=new Outer();

Outer.Inner i=o.new Inner();

i.m1();\*/

Outer.Inner i=**new** Outer().**new** Inner();

**new** Outer().**new** Inner().m1();

}

}

**2.Accessing Inner class code from instance area Of Outer class:**

**Ans:**

**publicclass** Outer {

**class** Inner{

**publicvoid** m1(){

System.*out*.println("inner classes method nnr");

}

}

**publicvoid** m2(){

Inner i=**new** Inner();

i.m1();

}

**publicstaticvoid** main(String[] args) {

Outer o=**new** Outer();

o.m2();

}

}

**3.Accessing Inner class code from outside of outer class:**

**Ans:**

**publicclass** Outer {

**class** Inner{

**publicvoid** m1(){

System.*out*.println("accessing from instance method nnr");

}

}

}

**class** Test{

**publicstaticvoid** main(String ar[]){

Outer o=**new** Outer();

Outer.Inner i=o.**new** Inner();

i.m1();

}

}

**4.From the inner class we can access all members of outer class(both static & non-static)directly.**

**Ans:publicclass** Outer {

**staticint***x*=10;

**int**y=44;

**finalint**h=22;

**class** Inner{

**publicvoid** m1(){

System.*out*.println(*x*);

System.*out*.println(y);

System.*out*.println(h);

System.*out*.println("accessing from instance method nnr");

}

}

**publicstaticvoid** main(String[] args) {

**new** Outer().**new** Inner().m1();

}

}

* With in the inner class this always pointing to current inner class object.
* To refer current outer class object we have to use “outerclassname.this”

**publicclass** Outer {

**int**x=44;

**finalint**h=22;

**class** Inner{

**int**x=11;

**publicvoid** m1(){

System.*out*.println(**this**.x);

System.*out*.println(Outer.**this**.x);

System.*out*.println("accessing from instance method nnr");

}

}

**publicstaticvoid** main(String[] args) {

**new** Outer().**new** Inner().m1();

}

}

**Applicable modifiers for outer 7 inner classes:**

**Outer**

Public,default,final,abstract,,strictfp

**Inner**

Outer+private,protected,static

**2.Method Local Inner classes:**

* We can declare a class inside a method such type of classes are called “Method Local Inner classes”.
* The main purpose of method local inner class is to define method specific functionality.
* Outside of the method we can’t access method local inner classes.

**publicclass** Outer {

**publicvoid** m1(){

**class** Methodinner{

**publicvoid** sum(**int** x,**int** y){

System.*out*.println("sum is"+(x+y));

}

}

Methodinner mi=**new** Methodinner();

mi.sum(10,6);

mi.sum(20,20);

mi.sum(60,40);

}

**publicstaticvoid** main(String[] args) {

**new** Outer().m1();

}}

3)**Annanymous Inner classes:**

* Sometimes we can declare a class without name also.such type of nameless inner classes are called anonymous inner classes
* There three types of anonymous inner classes
* Annonymous inner class that extends a class
* Annonymous inner class that implements an interface.
* Annonymous inner class that defined inside method arguments.

1.

**Publicclass** Popcorn {

**publicvoid** taste1(){

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

}

}

**class** Test{

**publicstaticvoid** main(String[] args) {

Popcorn p=**new** Popcorn()

{

**publicvoid** taste1()

{

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

}

};

p.taste1();

Popcorn p1=**new** Popcorn();

p1.taste1();

}

}

* The internal class name generated for anonymous inner class is “Test$1.class”

2. **publicclass** Popcorn {

**publicstaticvoid** main(String[] args) {

Runnable p=**new** Runnable() {

@Override

**publicvoid** run() {

**for**(**int** i=0;i<10;i++){

System.*out*.println("child thread");

}

}

};

Thread t=**new**Thread(p);

t.start();

**for**(**int** i=0;i<10;i++){

System.*out*.println("main thread");

}

}

}

3.**package** com.nnr.annonymousclass;

**publicclass** Popcorn {

**publicstaticvoid** main(String[] args) {

**new** Thread(**new** Runnable(){

@Override

**publicvoid** run() {

**for**(**int** i=0;i<10;i++){

System.*out*.println("child thread");

}

}

}).start();

**for**(**int** i=0;i<10;i++){

System.*out*.println("main thread");

}

}}

Annonymous can implement only one interface at time

**4.Static nested classes:**

* Sometimes we can declare inner class with static modifier such type of inner classes are called “static nested classes”

**publicclass** Popcorn {

**staticclass** Nested{

**publicstaticvoid** main(String[] args) {

System.*out*.println("static nested class main");

}

}

**publicstaticvoid** main(String[] args) {

System.*out*.println("outer class main");

}

}

**Java.lang package:**

The commonly used classes in lang

* Object
* String
* StringBuffer
* StringBuilder
* Wrapper classes(autoboxing & Auto unboxing)

**Object:**

**1.toString():**

public string toString():

Ans:

* we can use this method to find representation of an object
* whenever we are trying to print any object reference internally toString() will be executed

public String toString()

{

Return getclass().getName()+”@”+Integer.toHexString(hashcode());

}

In string,stringbuffer&in all wrapper classes toString() is overridden to return proper string form.Hence ,it is highly recommended to override toString() in our class also

**2.hashCode():**

* For every object jvm will assign one unique id which is nothing but hashcode.
* Jvm uses hashcode will saving object into hashtable or hashset or hashMap

**3.equals():**

* We can use equals() to check equality of two objects.

Public Boolean equals(Object o)

* If two references pointing to the same object then only .equals() returns true.this behavoiur is exactly same as == operator.
* If you want to perform content comparision instead of references comparision we have to override .equals() in our class.
* Whenever we are overriding .equals() we have to consider the following things
* In the case of diff type of objects(heterogeneous) equals() should return false but not classcastexception
* If we are passing null argument our .equals() should return false but not a NullpointerException

If(name1.equals(name2)&&rollno1==rollno2)

{

Return true;

}

Else{

Return false;

}catch(CCE e)

{

Return false;

}

Catch(Npe e){

Return false;}

**Differrence b/w == operator & .equals():**

**==operator .equals()**

1.It is an operator applicable for both 1.It is a method applicable only for

Primitives & object references. Object references but not for primitives.

2.in case of object references == 2.By default .equals() present in

Operator is always meant for references object class is also ment for

Comparision.If two ref erences pointing reference comparision only.

To the same object then only == operator

Returns true.

3.we can’t override == operator for 3.we can override .equals() for

Content comparision. Content comparision.

4.In case of heterogeneous type 4. In case of heterogeneous object

Objects == operator causes compiletime .equals() simply return false & we

Error saying incompatible types won’t get any compiletime or

Runtime error.

* In string, class .equals() is overriden for content comparision.
* In string buffer class .equals() is not overridden for content comparision hence object class .equals() got executed which is meant for reference comparision
* In wrapper class .equals() is overridden for content comparision

**Clone**():

* The process of creating exactly duplicate objects is called cloning.
* The main objective of cloning is to maintain backup.
* We can call clone() only on cloneable objects.
* An object is said to clonable iff the corresponding class implements clonable interface.clonable interface presently java.lang package & doesn’t contain any methods.It is a marker interface.

**DEEP cloning &shallow cloning:**

* The process of creating just duplicate reference variable but not duplicate object is called sallow cloning.
* The process of creating exactly duplicate independent objects is by default considered as deep cloning

Ex:Test t1=new Test();

Test t2=t1;//shallow cloning

T1 t2

**shallowcloning**

Test t3=(Test)t1.clone();//Deep cloning

T3

**Deepcloning**

* By default cloning means deep cloning

**OOPS Interview Questions:**

* **Constructor Interview Questions:**

**1.What is constructor?**

**Ans:**The main objective of the constructor is to perform initialization for the newly created object.

**2.What is the purpose of the default constructor?**

**3.Does Constructor returns any value?**

**Ans:**Return type concept is not applicable for constructor even void also.

-By mistake if we declare return type for the constructor we won't get any compiletime or run time errors.becz compiler treats it as a method.private ,public,protected,default these modifiers applicable for constructors.

**4.Is Constructor inherited?**

**Ans:**Inheritance & overriding concepts are not applicable for constructors.

-Every class in java including abstract class also can contain constructor.But inerface can't have the constructor

**5.Can we make a constructor final?**

**Ans:**

* Exception handling in Constructor is possible?
* Can we access static data in constructor.

**Abstraction Interview Questions:**

**1.What is abstraction?**

**Ans**:Hiding internal implementation details & just highylate the set of services what we are offering is called "Abstraction".

**ex**:by bank atm machine,Bank people highlite the set of services what they are offering without internal implementation this concept is nothing but abstraction.

* -By using interfaces & abstract classes we can achieve abstraction.
* -we can achieve security as no one is allowed to know our internal implementation.

**2.What is the difference between abstraction and encapsulation?**

Ans:Abstraction:same as above

**Encapsulation:**Encapsulating data & corresponding methods(behaviour) into single module is called "encapsulation".

**3.What is abstract class?**

**Ans**:If we are talking about implementation but not completely(just partially implementation) then we should go for abstract class.

ex:GenericServlet,HttpServlet.

* -An abstract class is a class that contains 0 or more abstract methods.
* -An abstract class can contain instance varibles and concrete methods in addition to abstract methods.

**4.Can we declare a class as abstract and final also?**

**Ans**: No .Abstract class needs subclass. final keyword can't be created subclasses.

**5. How can you force your programmers to implement only the features of your class?**

Ans:By writing an abstract class or an interface.

**6.Can there be any method which is abstract without abstract class?**

**Ans:**yes

7.**Can we use abstract and final both with a method?**

**Ans:**No

**8.Is it possible to instantiate the abstract class?**

**Ans:**

* we can't create an object to abstract class.
* -we can create a reference of abstract class type.
* -The reference of abstract class can be used to refer to objects of its subclass.

**9.What is interface?**

**Ans:**

* If we don't know any thing about implementation Just we have requirement specification.Then we should go for interface.
* --An interface is specification of method prototypes.

**Ex**:Servlet

**10.Why the methods of interfaces are public and abstract by default?**

**Ans:**

* By using public ,the method should be available to the third party vendors to provide implementation.
* By using abstract,Method implementation is left for third party vendors.

**11.what is concrete class?**

**Ans:**we are talking about implementation completely & ready to provide service.then we should go for Concrete class.

Ex:our own Servlet

**12.Can you declare an interface method static?**

**Ans:No**

**13.Can an interface be final?**

**14.What is marker interface?**

**Ans:**If an interface won't contain any method that interface is called Marker interface.

ex:Serializable,clonable

**15.What is the difference between abstract class and interface?**

**Ans:**

**Interface Abstract**

1.If we don't know any about implementation. 1.If we are talking about implementation but

just we have requirement specification.then we not completely (partial) then we should go

should go for interface. for abstract.

2.by default every method public abstract. 2.Here need not be p&ab.we can take

concrete method also.

3.here we can't take protected,static,final,private 3.here we can take any modifiers

synchroniged.

4.Every variable in interface,public static final by 4.here need not be p f s.

default wheather we are declare or not.

5.Inside interface we can't take instance & static 5.we can take here.

blocks.

6.we can't take constructor. 6.we can take

**16.Can we define private and protected modifiers for variables in an interface?**

**Ans:No**

**17.When can an object reference be cast to an interface reference**?

**Inheritance Interview Questions:**

**1.What is this in java?**

**Ans**:

* this is a keyword that refers to the object of the class where it is used.
* When an object is created to class,a default reference is also created internally to the object.this default reference is nothing but 'this'.so,'this' refer to all the things of the present object.

ex.this.x=x

this(55)\\call parameterized constructor

this.access()\\call present class method

**2.What is inheritance?**

**Ans:** Deriving new classes from existing classes such that the new classes acquire all the features of existing classes is called inheritance.

-By using extend keyword we can implement Is-A relationship.

-Main advantages is reusability of the code.

**3.Which class is the super class for every class?**

**Ans:**Object class

**4.Why multiple inheritance is not supported in java?**

**5.What is composition?**

**Ans:**In the case of composition,whenever container object is destroyed all contained objects will be destroyed automatically.i.e container and contained objects having strong association which is nothing but composition.it is a strong assocuation

Ex:University is compossed of several departments.

-whenever we are closing university automatically all departments will be closed.

**6.What is the difference between the aggregation and composition?**

**composition :** same as above

**Aggregation:**In the case of Aggregation,whenever container object is destroyed .there is no gauranty of destruction of contained objects.It is a weak assocition which is nothing but aggregation.

Ex:Several professors work in departments

**7.Why java does not support pointers?**

**Ans:**

**8.What is super in java?**

**Ans:**We always create an object to sub class in inheritance. some times, the super class members & sub class members may have the same names.in that case by default only sub class members are accessible. super keyword we can access the super class varibles & super class methods

-we want to access super class instance variable and super class method directly in sub class using super keywords.

**9.Can we use this () and super() both in a constructor?**

Ans:

**super():**When sub class object is created,first of all the super class default constructor is called and then only the sub class constructor is called.we take parameterized constructor in the super class.This not available to sub class by default. so it is should be called by using super().

**10.What is Object Cloning?**

**Ans:**Creating exact copy of an existing object is called 'cloning'

**Static key word Interview Questions:**

1.What is static variable?

Ans:A static variable (class variable) is a variable whose single copy in memory is shared by all objects.

**2.What is differrence b/w instance & static variable?**

**Ans:**

**instance variable static variable**

1**.**Aninstance variable is a variable whose seperate 1.A static variable

(class variable) is copy is availble to each object. a variable whose single copy in

memory is shared by all objects.

**2.What is static method?**

**Ans**:Static methods are methods which do not act upon tthe instance variable of a class.static method declared as static.

-The reason why static methods can not act on instance variables is that the Jvm first executes the static methods and then only it creates the objects.Since the objects are not available at the time of calling the staic methods ,The instance variables are not available.

**3.What are the instance methods?**

**Ans**:instance methods act on the instance variables of the class .by using object name we can call the instance methods.

There are two types of instance methods.

1.Accessor methods

2.Mutator methods

**4.Why main method is static?**

**Ans:**the jvm executes first of all any static blocks in the java program.then it executes static methods(main method) and then it creates any objects by the program .finally it executes the instance methods.

**5.What is static block?**

**Ans**:static block is a block of statements declared as a static.

-Jvm first executes static blocks.Jvm first goes to static block even before it looks the main() in the program.If main() is not found it will displays an error(nosuchmethoderror).

**6.Can we execute a program without a main method?**

**Ans:**yes,it is possible to run java program by using static block.

**7.What happens if the static modifier is removed from the main method?**

**Ans:**Main method is not static in class , please define the main method

**8.What is the difference between the static method and instance method?**

**Method overloading Interview Questions:**

**1.What is method overloading?**

**Ans:**Writing two or more methods in the same class in such a way that each method has same names but with differrent arguments.

2.Why method overloading is not possible by change the return type in java?

3.Can we overload main () method?

***Method overriding Interview Questions:***

**1.What is method overriding?**

Ans:Writing two or more methods win super and sub classes such that the methods have same name and same signature is called MO.

Overriding is also known as "runtime polymorphism" or "dynamic polymorphism" or "late binding".

**2.Can we override a static method**?

ans:No

**What is co-variant return type?**

Ans:co-variant return type concept is applicable only for Object type but not for primitive types.

-In overriding return type must be matched,but this rule is apllicable untill 1.4 version ,from 1.5 version onwards Co-variant return types are allowed.