1)WHAT IS JVM AND EXPLAIN ME THE JAVA MEMORY ALLOCATION

ANS: JVM is short for ***Java Virtual Machine***. JVM is an abstract computing machine, or virtual machine. It is a platform-independent execution environment that converts Java bytecode into machine language and executes it. Most programming languages compile source code directly into machine code that is designed to run on a specific operating system, such as Windows or UNIX.

The JVM memory consists of the following segments:

-Heap Memory, which is the storage for Java objects

-Non-Heap Memory, which is used by Java to store loaded classes and other meta-data

-JVM code itself, JVM internal structures, loaded profiler agent code and data, etc.

2)WHAT IS POLYMORPHISM AND ENCAPSULATION?

ANS:Polymorphism in java is a concept by which we can perform a single action by different ways. So polymorphism means many forms.

There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism. We can perform compile time polymorphism in java by method overloading and run time polymorphism by method overriding.

Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

```
class Bank{

int getRateOfInterest(){

return 0;

}

}


class SBI extends Bank{

int getRateOfInterest(){return 8;}

}
```

```
class ICICI extends Bank{

int getRateOfInterest(){return 7;}

}

class AXIS extends Bank{

int getRateOfInterest(){return 9;}

}


class Test3{

public static void main(String args[]){

Bank b1=new SBI();

Bank b2=new ICICI();

Bank b3=new AXIS();

System.out.println("SBI Rate of Interest: "+b1.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+b2.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+b3.getRateOfInterest());

}

}
```

Output:

```
SBI Rate of Interest: 8

ICICI Rate of Interest: 7

AXIS Rate of Interest: 9
```

ENCAPSULATION:Encapsulation in java is a

We can create a fully encapsulated class in java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

The Java Bean class is the example of fully encapsulated class.

Advantage of Encapsulation in java

By providing only setter or getter method, you can make the class read-only or write-only.

```
package com.javatpoint;

public class Student{

private String name;


public String getName(){

return name;

}

public void setName(String name){

this.name=name

}

}

package com.javatpoint;

class Test{

public static void main(String[] args){

Student s=new Student();

s.setname("vijay");

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

}

}
```

3)WHAT IS METHOD OVERLOADING AND METHOD OVER RIDING?

ANS:If a class have multiple methods by same name but different parameters, it is known as Method Overloading.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Advantage of method overloading?

Method overloading increases the readability of the program.

Different ways to overload the method

There are two ways to overload the method in java

-By changing number of arguments

-By changing the data type

In java, Methood Overloading is not possible by changing the return type of the method.

1)Example of Method Overloading by changing the no. of arguments

In this example, we have created two overloaded methods, first sum method performs addition of two numbers and second sum method performs addition of three numbers.

```
class Calculation{
  void sum(int a,int b){System.out.println(a+b);}
  void sum(int a,int b,int c){System.out.println(a+b+c);}


  public static void main(String args[]){
  Calculation obj=new Calculation();
  obj.sum(10,10,10);
  obj.sum(20,20);


  }
}
```

Test it Now

Output:30

```
      40
```

2)Example of Method Overloading by changing data type of argument

In this example, we have created two overloaded methods that differs in data type. The first sum method receives two integer arguments and second sum method receives two double arguments.

```
class Calculation2{
  void sum(int a,int b){System.out.println(a+b);}
  void sum(double a,double b){System.out.println(a+b);}

  public static void main(String args[]){
  Calculation2 obj=new Calculation2();
  obj.sum(10.5,10.5);
  obj.sum(20,20);

  }
}
```

Test it Now

```
Output:21.0
      40
```

METHOD OVERRIDING:If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding

Method overriding is used to provide specific implementation of a method that is already provided by its super class.

Method overriding is used for runtime polymorphism

Rules for Java Method Overriding:

-method must have same name as in the parent class

-method must have same parameter as in the parent class.

-must be IS-A relationship (inheritance).

Understanding the problem without method overriding

Let's understand the problem that we may face in the program if we don't use method overriding.

```
class Vehicle{

  void run(){System.out.println("Vehicle is running");}

}

class Bike extends Vehicle{



  public static void main(String args[]){

  Bike obj = new Bike();

  obj.run();

  }

}
```

Test it Now

Output:Vehicle is running

Problem is that I have to provide a specific implementation of run() method in subclass that is why we use method overriding.

Example of method overriding

In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method is same and there is IS-A relationship between the classes, so there is method overriding.

```
class Vehicle{

void run(){System.out.println("Vehicle is running");}

}

class Bike2 extends Vehicle{

void run(){System.out.println("Bike is running safely");}


public static void main(String args[]){

Bike2 obj = new Bike2();

obj.run();

}
```

Output:Bike is running safely

4)WHY STRING IS IMMUTABLE

ANS: Java runtime allocates a fixed amount of memory equal to the length of the string whwn a string object is created,so its not possible to change the memory allocated to the string

String is designed to be immutable for the sake of efficiency and security.

When we create a string in java like String s1="hello"; then an object will be created in string pool(hello) and s1 will be pointing to hello.Now if again we do String s2="hello"; then another object will not be created but s2 will point to hello because JVM will first check if the same object is present in string pool or not.If not present then only a new one is created else not.

5)What is the difference between String and String buffer?

ANS: String is immutable (once created can not be changed)object . The object created as a String is stored in the Constant String Pool .

Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.

String once assigned can not be changed.

String demo = " hello " ;

// The above object is stored in constant string pool and its value can not be modified.

demo="Bye" ; //new "Bye" string is created in constant pool and referenced by the demo variable

// "hello" string still exists in string constant pool and its value is not overrided but we lost reference to the "hello"string

StringBuffer

StringBuffer is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap . StringBuffer has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe .

Due to this it does not allow two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .

6)Difference between Array and Arraylist?

	Array	**ArrayList**
Resizable	**No**	**Yes**
Primitives	**Yes**	**No**
Iterating values	**for, for each**	**Iterator , for each**
Length	**length variable**	**size method**
Performance	**Fast**	**Slow in comparision**
Multidimensional	**Yes**	**No**
Add Elements	**Assignment operator**	**add method**

7)What is the difference between hash map and Hash table?

HashMap	**Hashtable**

Synchronized	**No**	**Yes**
Thread-Safe	**No**	**Yes**
Null Keys and Null values	**One null key ,Any null values**	**Not permit null keys and values**
Iterator type	**Fail fast iterator**	**Fail safe iterator**
Performance	**Fast**	**Slow in comparison**
Superclass and Legacy	**AbstractMap , No**	**Dictionary , Yes**

8. What is a vector in Java?

ANS:Vector implements List Interface. Like ArrayList it also maintains insertion order but it is rarely used in non-thread environment as it is synchronized and due to which it gives poor performance in searching, adding, delete and update of its elements.

Three ways to create vector class object:

Method 1:

Vector vec = new Vector();

Method 2:

Vector object= new Vector(int initialCapacity)

Vector vec = new Vector(3);

Syntax:

Vector object= new vector(int initialcapacity, capacityIncrement)

Vector vec= new Vector(4, 6)

9. What is set in java?

ANS:A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited. Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ. Two Set instances are equal if they contain the same elements.

10. What is an abstract class?

ANS:An abstract class is a class that is declared abstract

Abstract classes cannot be instantiated

Abstract classes can be subclassed

It may or may not include abstract methods

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class

Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods. Let's look at an example of an abstract class, and an abstract method.

```
public abstract Animal

{

  public void eat(Food food)

  {

     // do something with food....

  }


  public void sleep(int hours)

  {

     try

        {

                // 1000 milliseconds * 60 seconds * 60 minutes * hours

                Thread.sleep ( 1000 * 60 * 60 * hours);

        }

        catch (InterruptedException ie) { /* ignore */ }
  }

  public abstract void makeNoise();
}
```

Note that the abstract keyword is used to denote both an abstract method, and an abstract class. Now, any animal that wants to be instantiated (like a dog or cow) must implement the makeNoise method - otherwise it is impossible to create an instance of that class. Let's look at a Dog and Cow subclass that extends the Animal class.

```
public Dog extends Animal
{
  public void makeNoise() { System.out.println ("Bark! Bark!"); }
}


public Cow extends Animal
{
  public void makeNoise() { System.out.println ("Moo! Moo!"); }
}
```

11. What is an interface?

ANS:An interface is similar to a class in the following ways:

An interface can contain any number of methods.

Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including:

You cannot instantiate an interface.

An interface does not contain any constructors.

All of the methods in an interface are abstract.

An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.

An interface is not extended by a class; it is implemented by a class.

An interface can extend another interfaces.

12. Why Java is Platform independent?Java language is platform independent but not JVM

ANS:Here, platform here refers to the operating system. Once the java program is compiled, it generates the bytecode. And this bytecode is portable, and hence could be run anywhere. This bytecode is in hexadecimal format,. This format is same everywhere, so when the program is run , generated byte code translates it according to the host machine. Here JVM is responsible for interpretation of bytecode.

Platform independent means that we can write and compile that java code in one platform(eg windows) and can execute the class in any other supported platform (eg Linux,Solaris etc.).So java is a platform independent,But JVM is not platform independent. JVM's are platform specific run time implementation provided by vendor.

13.What are Access Modifiers?

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are:

Visible to the package. the default. No modifiers are needed.

Visible to the class only (private).

Visible to the world (public).

Visible to the package and all subclasses (protected).

14)What are java exceptions? Give me an example

ANS:What is exception

Dictionary Meaning: Exception is an abnormal condition.

In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Types of Exception

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:

Checked Exception

Unchecked Exception

Error

Difference between checked and unchecked exceptions

1) Checked Exception

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

2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

3) Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

15. What is the difference between throws and throwable?

ANS:throw In Java :

throw is a keyword in java which is used to throw an exception manually. Using throw keyword, you can throw an exception from any method or block. But, that exception must be of type java.lang.Throwable class or it's sub classes. Below example shows how to throw an exception using throw keyword.

```
class ThrowAndThrowsExample

{
   void method() throws Exception

   {

      Exception e = new Exception();



      throw e;            //throwing an exception using 'throw'

   }

}
```

throws In Java :

throws is also a keyword in java which is used in the method signature to indicate that this method may throw mentioned exceptions. The caller to such methods must handle the mentioned exceptions either using try-catch blocks or using throws keyword. Below is the syntax for using throws keyword.

```
return_type method_name(parameter_list) throws exception_list

{

    //some statements

}
```

Below is the example which shows how to use throws keyword.

```
class ThrowsExample

{

   void methodOne() throws SQLException

   {

      //This method may throw SQLException

   }
   void methodTwo() throws IOException

   {

      //This method may throw IOException

   }



   void methodThree() throws ClassNotFoundException

   {

      //This method may throw ClassNotFoundException

   }

}
```

Throwable In Java :

Throwable is a super class for all types of errors and exceptions in java. This class is a member of java.lang package. Only instances of this class or it's sub classes are thrown by the java virtual machine or by the throw statement. The only argument of catch block must be of this type or it's sub classes. If you want to create your own customized exceptions, then your class must extend this class.

Below example shows how to create customized exceptions by extending java.lang.Throwable class.

```
class MyException extends Throwable

{

        //Customized Exception class

}
```

```
class ThrowAndThrowsExample

{

   void method() throws MyException

   {

      MyException e = new MyException();



      throw e;

   }

}
```

16. What is the difference between Error and exception?

ANS:Errors	**Exceptions**
Errors in java are of type java.lang.Error.	**Exceptions in java are of type java.lang.Exception.**
All errors in java are unchecked type.	**Exceptions include both checked as well as unchecked type.**
Errors happen at run time. They will not be known to compiler.	**Checked exceptions are known to compiler where as unchecked exceptions are not known to compiler because they occur at run time.**

It is impossible to recover from errors.   You can recover from exceptions by handling them through try-catch blocks.

Errors are mostly caused by the environment in which application is running.
Exceptions are mainly   caused by the application itself.

Examples :

java.lang.StackOverflowError, java.lang.OutOfMemoryError Examples :

Checked Exceptions : SQLException, IOException

Unchecked Exceptions : ArrayIndexOutOfBoundException, ClassCastException, NullPointerException

18. What are collection APIs, give me an example

ANS:The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces.

Example of classes: HashSet, HashMap, ArrayList, LinkedList, TreeSet and TreeMap.
Example of interfaces: Collection, Set, List and Map.

19. What is the difference between final and finally?

ANS:Final:- It is used in the following cases:

If the final keyword is attached to a variable then the variable becomes constant i.e. its value cannot be changed in the program.

If a method is marked as final then the method cannot be overridden by any other method.

If a class is marked as final then this class cannot be inherited by any other class.

If a parameter is marked with final it becomes a read only parameter.

Finally:-

If an exception is thrown in try block then the control directly passes to the catch block without executing the lines of code written in the remainder section of the try block. In case of an exception we may need to clean up some objects that we created. If we do the clean-up in try block, they may not be executed in case of an exception. Thus finally block is used which contains the code for clean-up and is always executed after the try ...catch block.

20. Will java supports multiple inheritance?

ANS:Because interfaces specify only what the class is doing, not how it is doing it.

The problem with multiple inheritance is that two classes may define different ways of doing the same thing, and the subclass can't choose which one to pick.

21. What are the different types of interface?

Ans List, set, Queue

22. What are wrapper class? Give me an example

ANS:Java wrapper classes are the Object representation of eight primitive types in java. All the wrapper classes in java are immutable and final. Java 5 autoboxing and unboxing allows easy conversion between primitive types and their corresponding wrapper classes.

23. What is boxing and unboxing in Java? Explain with an example

ANS:The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So java programmer doesn't need to write the conversion code.

Advantage of Autoboxing and Unboxing:

No need of conversion between primitives and Wrappers manually so less coding is required.

Simple Example of Autoboxing in java:

```
class BoxingExample1{

  public static void main(String args[]){

    int a=50;
```

```
        Integer a2=new Integer(a);//Boxing


        Integer a3=5;//Boxing


        System.out.println(a2+" "+a3);
 }
}
```

Test it Now

```
Output:50 5
```

download this example

Simple Example of Unboxing in java:

The automatic conversion of wrapper class type into corresponding primitive type, is known as Unboxing. Let's see the example of unboxing:

```
class UnboxingExample1{
  public static void main(String args[]){
    Integer i=new Integer(50);
        int a=i;


        System.out.println(a);
 }
}
```

Test it Now

Output:50

24. Explain for each loop

ANS:The for-each loop introduced in Java5. It is mainly used to traverse array or collection elements. The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Advantage of for-each loop:

It makes the code more readable.

It elimnates the possibility of programming errors.

Syntax of for-each loop:

```
for(data_type variable : array | collection){}
```

Simple Example of for-each loop for traversing the array elements:

```
class ForEachExample1{
  public static void main(String args[]){
   int arr[]={12,13,14,44};

   for(int i:arr){
     System.out.println(i);
   }

 }
}
```

Test it Now

```
Output:12
       13
       14
       44
```

Simple Example of for-each loop for traversing the collection elements:

```
import java.util.*;
class ForEachExample2{
  public static void main(String args[]){
   ArrayList<String> list=new ArrayList<String>();
   list.add("vimal");
   list.add("sonoo");
   list.add("ratan");

   for(String s:list){
     System.out.println(s);
   }

 }
}
```

Test it Now

```
Output:vimal
       sonoo
       rattan
```

25. What are iterators, explain with an example

ANS:Iterator in Java is nothing but a traversing object, made specifically for Collection objects like List and Set. we have already aware about different kind of traversing methods like for-loop ,while loop,do-while,for each lop etc,they all are index based traversing but as we know Java is purely object oriented language there is always possible ways of doing things using objects so Iterator is a way to traverse as well as access the data from the collection.

26. How do you access Private variables in different class ?

ANS:

The correct way to access a private variable from another class is with getter and setter methods. Otherwise, you should have made that variable public.

That is:

```
// getter

public JLabel getMainLabel() {

   return mainlabel;

}


// setter

public void setMainLabel(JLabel mainLabel) {

   this.mainlabel = mainLabel;
```

28)CONSTRUCTOR OVERLOADING:

ANS Just like in the case of method overloading you have multiple methods with the same name but different signature, in Constructor overloading you have multiple constructors with a different signature with the only difference that Constructor doesn't have a return type in Java. That constructor will be called as an overloaded constructor . Overloading is also another form of polymorphism in Java which allows having multiple constructors with a different name in one Class in java.

29)With out using sync key word how do you perform synchronization ?

ANS:No need for synchronization at all if you don't have mutable state.

No need for synchronization if the mutable state is confined to a single thread. This can be done by using local variables or java.lang.ThreadLocal.

You can also use built-in synchronizers. java.util.concurrent.locks.ReentrantLock has the same functionality as the lock you access when using synchronized blocks and methods, and it is even more powerful.

30) What is super keyword ? when and where do you use it ?

ANS:In case of inheritance super keyword is used to refer to the base class.It is used when a member of the base class has the same name as the member in the sub class.In such cases super keyword can be used in subclass to refer to the member of the base class.

It can also be used to invoke the constructor of base class from the constructor of the sub class.