1. What is Stack Memory?

Stack memory is used to store local variables and function calls while heap memory is used to store objects in Java. Stack memory is always referenced in Last-In-First-Out order. Local variables are created in the stack.

Each Thread in Java has their own stack which can be specified using -Xss JVM parameter

2. What is Heap Memory?

Heap is a section of memory which contains Objects and may also contain reference variables. Instance variables are created in the heap.

Heap size of Java program can be specified using JVM option -Xms and -Xmx where -Xms is starting size of the heap and -Xmx is a maximum size of java heap.

3. What is difference between Static and Final keyword?

Static Keyword:

\* Static keyword is used to do better memory management.

\* You can use static keyword with variable, method, class.

\* If you declare method using static keyword then you don't need to create an object to call that method.

\* If you declare variable as a static then only once memory is created for that variable in class area and not every time when object is created. e.g. in student database you can declare "collage name" as because it common for all students.

Final Keyword:

\* Final Keyword is used to restrict the user. It means:

\* If you declare variable as static then you cannot change its value.

\* If you declare method as static then you cannot override that method.

\* If you make any class as final, you cannot extend it.

Q #1) What is JAVA?

Answer: Java is a high-level programming language and is platform-independent.

Java is a collection of objects. It was developed by Sun Microsystems. There are a lot of

applications, websites, and games that are developed using Java.

Q #2) What are the features in JAVA?

Answer: Features of Java are as follows:

 OOPs concepts

o Object-oriented

o Inheritance

o Encapsulation

o Polymorphism

o Abstraction

 Platform independent: A single program works on different platforms without any modification.

 High Performance: JIT (Just In Time compiler) enables high performance in Java.

JIT converts the bytecode into machine language and then JVM starts the execution.

 Multi-threaded: A flow of execution is known as a Thread. JVM creates a thread

which is called the main thread. The user can create multiple threads by extending

the thread class or by implementing the Runnable interface.

Q #3) How does Java enable high performance?

Answer: Java uses Just In Time compiler to enable high performance. It is used to convert

the instructions into bytecodes.

Q #4) Name the Java IDE’s?

Answer: Eclipse and NetBeans are the IDE's of JAVA.

Q #5) What do you mean by Constructor?

Answer: Constructor can be explained in detail with enlisted points:

 When a new object is created in a program a constructor gets invoked corresponding

to the class.

 The constructor is a method which has the same name as the class name.

 If a user doesn’t create a constructor implicitly a default constructor will be created.

 The constructor can be overloaded.

 If the user created a constructor with a parameter then he should create another

constructor explicitly without a parameter.

Q #6) What is meant by the Local variable and the Instance variable?

Answer:

Local variables are defined in the method and scope of the variables existed inside the method itself.

Instance variable is defined inside the class and outside the method and the scope of the variables exists throughout the class.

Q #7) What is a Class?

Answer: All Java codes are defined in a Class. It has variables and methods.

Variables are attributes which define the state of a class.

Methods are the place where the exact business logic has to be done. It contains a set of statements (or) instructions to satisfy the particular requirement.

Example:

public class Addition{ //Class name declaration

int a = 5; //Variable declaration

int b= 5;

public void add(){ //Method declaration

int c = a+b;

}

}

Q #8) What is an Object?

Answer: An instance of a class is called an object. The object has state and behavior.

Whenever the JVM reads the “new()” keyword then it will create an instance of that class.

Example:

public class Addition{

public static void main(String[] args){

Addion add = new Addition();//Object creation

}

}

The above code creates the object for the Addition class.

Q #9)What are the OOPs concepts?

Answer: OOPs concepts include:

 Inheritance

 Encapsulation

 Polymorphism

 Abstraction

 Interface

Q #10) What is Inheritance?

Answer: Inheritance means one class can extend to another class. So that the codes can

be reused from one class to another class. The existing class is known as the Super class

whereas the derived class is known as a sub class.

Example:

Super class:

public class Manupulation(){

}

Sub class:

public class Addition extends Manipulation(){

}

Inheritance is applicable for public and protected members only. Private members can’t be

inherited.

Q #11) What is Encapsulation?

Answer: Purpose of Encapsulation:

 Protects the code from others.

 Code maintainability.

Example:

We are declaring ‘a' as an integer variable and it should not be negative.

public class Addition(){

int a=5;

}

If someone changes the exact variable as “a = -5” then it is bad.

In order to overcome the problem we need to follow the below steps:

 We can make the variable as private or protected one.

 Use public accessor methods such as set<property> and get<property>.

So that the above code can be modified as:

public class Addition(){

private int a = 5; //Here the variable is marked as private

}

The below code shows the getter and setter.

Conditions can be provided while setting the variable.

get A(){

}

set A(int a){

if(a>0){// Here condition is applied

.........

}

}

For encapsulation, we need to make all the instance variables as private and create setter

and getter for those variables. Which in turn will force others to call the setters rather than

access the data directly.

Q #12) What is Polymorphism?

Answer: Polymorphism means many forms.

A single object can refer to the super-class or sub-class depending on the reference type

which is called polymorphism.

Example:

Public class Manipulation(){ //Super class

public void add(){

}

}

public class Addition extends Manipulation(){ // Sub class

public void add(){

}

public static void main(String args[]){

Manipulation addition = new Addition();//Manipulation is reference type and Addition is reference type

addition.add();

}

}

Using Manipulation reference type we can call the Addition class “add()” method. This ability

is known as Polymorphism. Polymorphism is applicable for overriding and not for

overloading.

Q #13) What is meant by Method Overriding?

Answer: Method overriding happens if the sub-class method satisfies the below

conditions with the Super-class method:

 Method name should be the same

 The argument should be the same

 Return type also should be the same

The key benefit of overriding is that the Sub-class can provide some specific information

about that sub-class type than the super-class.

Example:

public class Manipulation{ //Super class

public void add(){

………………

}

}

Public class Addition extends Manipulation(){

Public void add(){

………..

}

Public static void main(String args[]){

Manipulation addition = new Addition(); //Polimorphism is applied

addition.add(); // It calls the Sub class add() method

}

}

addition.add() method calls the add() method in the Sub-class and not the parent class.

So it overrides the Super-class method and is known as Method Overriding.

Q #14) What is meant by Overloading?

Answer: Method overloading happens for different classes or within the same class.

For method overloading, sub-class method should satisfy the below conditions

with the Super-class method (or) methods in the same class itself:

 Same method name

 Different argument type

 May have different return types

Example:

public class Manipulation{ //Super class

public void add(String name){ //String parameter

………………

}

}

Public class Addition extends Manipulation(){

Public void add(){//No Parameter

………..

}

Public void add(int a){ //integer parameter

}

Public static void main(String args[]){

Addition addition = new Addition();

addition.add();

}

}

Here the add() method has different parameters in the Addition class is overloaded in the

same class as well as with the super-class.

Note: Polymorphism is not applicable for method overloading.

Q #15) What is meant by Interface?

Answer: Multiple inheritances cannot be achieved in java. To overcome this problem Interface concept is introduced.

An interface is a template which has only method declarations and not the method implementation.

Example:

Public abstract interface IManupulation{ //Interface declaration

Public abstract void add();//method declaration

public abstract void subtract();

}

 All the methods in the interface are internally public abstract void.

 All the variables in the interface are internally public static final that is constants.

 Classes can implement the interface and not extends.

 The class which implements the interface should provide an implementation for all

the methods declared in the interface.

public class Manupulation implements IManupulation{ //Manupulation class uses the interface

Public void add(){

……………

}

Public void subtract(){

…………….

}

}

Q #16) What is meant by Abstract class?

Answer: We can create the Abstract class by using the “Abstract” keyword before the class name. An abstract class can have both “Abstract” methods and “Non-abstract” methods that are a concrete class.

Abstract method:

The method which has only the declaration and not the implementation is called the abstract method and it has the keyword called “abstract”. Declarations are the ends with a semicolon.

Example:

public abstract class Manupulation{

public abstract void add();//Abstract method declaration

Public void subtract(){

}

}

 An abstract class may have a non- abstract method also.

 The concrete Subclass which extends the Abstract class should provide the

implementation for abstract methods.

Q #17) Difference between Array and Array List.

Answer: The Difference between Array and Array List can be understood from the

below table:Array Array List

Size should be given at the time of array declaration.

String[] name = new String[2]

Size may not be required. It changes the size dynamically.

ArrayList name = new ArrayList

To put an object into array we need to specify the index.

name[1] = “book”

No index required.

name.add(“book”)

Array is not type parameterized

ArrayList in java 5.0 are

parameterized.

Eg: This angle bracket is a type

parameter which means a list of String.

Q #18) Difference between String, String Builder, and String Buffer.

Answer:

String: String variables are stored in a “constant string pool”. Once the string reference

changes the old value that exists in the “constant string pool”, it cannot be erased.

Example:

String name = “book”;

Constant string pool

.

If the name-value has changed from “book” to “pen”.

Constant string pool

Then the older value retains in the constant string pool.

String Buffer:

 Here string values are stored in a stack. If the values are changed then the new

value replaces the older value.

 The string buffer is synchronized which is thread-safe.

 Performance is slower than the String Builder.

Example:

String Buffer name =”book”;

Once the name value has been changed to “pen” then the “book” is erased in the stack.

String Builder:

This is the same as String Buffer except for the String Builder which is not threaded safety

that is not synchronized. So obviously performance is fast.

Q #19) Explain about Public and Private access specifiers.

Answer: Methods and instance variables are known as members.

Public:

Public members are visible in the same package as well as the outside package that is for

other packages.

Public members in Class A are visible to Class B (same package) as well as Class C

(different packages).

Private:

Private members are visible in the same class only and not for the other classes in the same

package as well as classes in the outside packages.

Private members in class A are visible only in that class. It is invisible for class B as well as

class C.

Q #20) Difference between Default and Protected access specifiers.

Answer:

Default: Methods and variables declared in a class without any access specifiers are called

default.

Default members in Class A are visible to the other classes which are inside the package

and invisible to the classes which are outside the package.

So Class A members are visible to the Class B and invisible to the Class C.

Protected:

.

Protected is the same as Default but if a class extends then it is visible even if it is outside

the package.

Class A members are visible to Class B because it is inside the package. For Class C it is

invisible but if Class C extends Class A then the members are visible to the Class C even if it

is outside the package.

Q #21) Difference between HashMap and HashTable.

Answer: Difference between HashMap and HashTable can be seen below:

HashMap HashTable

Methods are not synchronized Key methods are synchronized

Not thread safety Thread safety

Iterator is used to iterate the values Enumerator is used to iterate the values

Allows one null key and multiple null values Doesn’t allow anything that is null

HashMap HashTable

Performance is high than HashTable Performance is slow

Q #22) Difference between HashSet and TreeSet.

Answer: Difference between HashSet and TreeSet can be seen below:

HashSet TreeSet

Inserted elements are in random order Maintains the elements in the sorted order

Can able to store null objects Couldn’t store null objects

Performance is fast Performance is slow

Q #23) Difference between Abstract class and Interface.

Answer: Difference between Abstract Class and Interface are as follows:

Abstract Class:

 Abstract classes have a default constructor and it is called whenever the concrete

subclass is instantiated.

 It contains Abstract methods as well as Non-Abstract methods.

 The class which extends the Abstract class shouldn’t require the implementation of

all the methods, only Abstract methods need to be implemented in the concrete subclass.

 Abstract class contains instance variables.

Interface:

 It doesn’t have any constructor and couldn’t be instantiated.

 The abstract method alone should be declared.

 Classes that implement the interface should provide the implementation for all the methods.

 The interface contains only constants.

Q #24) What is mean by Collections in Java?

Answer: Collection is a framework that is designed to store the objects and manipulate the

design to store the objects.

Collections are used to perform the following operations:

 Searching

 Sorting

 Manipulation

 Insertion

 Deletion

A group of objects is known as collections. All the classes and interfaces for collecting are available in Java util package.

Q #25) What are all the Classes and Interfaces that are available in the

collections?

Answer: Given below are the Classes and Interfaces that are available in

Collections:

Interfaces:

 Collection

 List

 Set

 Map

 Sorted Set

 Sorted Map

 Queue

Classes:

 Lists:

 Array List

 Vector

 Linked List

Sets:

 Hash set

 Linked Hash Set

 Tree Set

Maps:

 Hash Map

 Hash Table

 TreeMap

 Linked Hashed Map

Queue:

 Priority Queue

Q #26) What is meant by Ordered and Sorted in collections?

Answer:

Ordered: It means the values that are stored in a collection is based on the values that are

added to the collection. So we can iterate the values from the collection in a specific order.

Sorted: Sorting mechanisms can be applied internally or externally so that the group of objects sorted in a particular collection is based on the properties of the objects.

Q #27) Explain the different lists available in the collection.

Answer: Values added to the list is based on the index position and it is ordered by index position. Duplicates are allowed.

Types of Lists are:

a) Array List:

 Fast iteration and fast Random Access.

 It is an ordered collection (by index) and not sorted.

 It implements the Random Access Interface.

Example:

public class Fruits{

public static void main (String [ ] args){

ArrayList <String>names=new ArrayList <String>();

names.add (“apple”);

names.add (“cherry”);

names.add (“kiwi”);

names.add (“banana”);

names.add (“cherry”);

System.out.println (names);

}

}

Output:

[Apple, cherry, kiwi, banana, cherry]

From the output, Array List maintains the insertion order and it accepts the duplicates. But

not sorted.

b) Vector:

It is the same as Array List.

 Vector methods are synchronized.

 Thread safety.

 It also implements the Random Access.

 Thread safety usually causes a performance hit.

Example:

public class Fruit {

public static void main (String [ ] args){

Vector <String> names = new Vector <String> ( );

names.add (“cherry”);

names.add (“apple”);

names.add (“banana”);

names.add (“kiwi”);

names.add (“apple”);

System.out.println (“names”);

}

}

Output:

[cherry,apple,banana,kiwi,apple]

Vector also maintains the insertion order and accepts the duplicates.

c) Linked List:

 Elements are doubly linked to one another.

 Performance is slower than the Array list.

 Good choice for insertion and deletion.

 In Java 5.0 it supports common queue methods peek( ), Pool ( ), Offer ( ) etc.

Example:

public class Fruit {

public static void main (String [ ] args){

Linkedlist <String> names = new linkedlist <String> ( ) ;

names.add(“banana”);

names.add(“cherry”);

names.add(“apple”);

names.add(“kiwi”);

names.add(“banana”);

System.out.println (names);

}

}

Output:

[ banana,cherry,apple,kiwi,banana]

Maintains the insertion order and accepts the duplicates.

Q #28) Explain about Set and their types in a collection?

Answer: Set cares about uniqueness. It doesn’t allow duplications. Here “equals ( )”

method is used to determine whether two objects are identical or not.

a) Hash Set:

 Unordered and unsorted.

 Uses the hash code of the object to insert the values.

 Use this when the requirement is “no duplicates and don’t care about the order”.

Example:

public class Fruit {

public static void main (String[ ] args){

HashSet<String> names = new HashSet <=String>( ) ;

names.add(“banana”);

names.add(“cherry”);

names.add(“apple”);

names.add(“kiwi”);

names.add(“banana”);

System.out.println (names);

}

}

Output:

[banana, cherry, kiwi, apple]

It doesn’t follow any insertion order. Duplicates are not allowed.

b) Linked Hash set:

 An ordered version of the hash set is known as Linked Hash Set.

 Maintains a doubly-Linked list of all the elements.

 Use this when the iteration order is required.

Example:

public class Fruit {

public static void main (String[ ] args){

LinkedHashSet<String>; names = new LinkedHashSet <String>( ) ;

names.add(“banana”);

names.add(“cherry”);

names.add(“apple”);

names.add(“kiwi”);

names.add(“banana”);

System.out.println (names);

}

}

Output:

[banana, cherry, apple, kiwi]

It maintains the insertion order in which they have been added to the Set. Duplicates are

not allowed.

c) Tree Set:

 It is one of the two sorted collections.

 Uses the “Read-Black” tree structure and guarantee that the elements will be in

ascending order.

 We can construct a tree set with the constructor by using a comparable (or)

comparator.

Example:

public class Fruits{

public static void main (String[ ]args) {

Treeset<String> names= new TreeSet<String>( ) ;

names.add(“cherry”);

names.add(“banana”);

names.add(“apple”);

names.add(“kiwi”);

names.add(“cherry”);

System.out.println(names);

}

}

Output:

[apple, banana, cherry, kiwi]

TreeSet sorts the elements in ascending order. And duplicates are not allowed.

Q #29). Explain about Map and their types.

Answer: Map cares about the unique identifier. We can map a unique key to a specific

value. It is a key/value pair. We can search a value, based on the key. Like the set, Map

also uses the “equals ( )” method to determine whether two keys are the same or different.

Map is of following types:

a) Hash Map:

 Unordered and unsorted map.

 Hashmap is a good choice when we don’t care about the order.

 It allows one null key and multiple null values.

Example:

Public class Fruit{

Public static void main(String[ ] args){

HashMap<Sting,String> names =new HashMap<String,String>( );

names.put(“key1”,“cherry”);

names.put (“key2”,“banana”);

names.put (“key3”,“apple”);

names.put (“key4”,“kiwi”);

names.put (“key1”,“cherry”);

System.out.println(names);

}

}

Output:

{key2 =banana, key1=cherry, key4 =kiwi, key3= apple}

Duplicate keys are not allowed in Map.

It doesn’t maintain any insertion order and is unsorted.

b) Hash Table:

 Like the vector key, methods of the class are synchronized.

 Thread safety and therefore slows the performance.

 It doesn’t allow anything that is null.

Example:

public class Fruit{

public static void main(String[ ]args){

Hashtable<Sting,String> names =new Hashtable<String,String>( );

names.put(“key1”,“cherry”);

names.put(“key2”,“apple”);

names.put(“key3”,“banana”);

names.put(“key4”,“kiwi”);

names.put(“key2”,“orange”);

System.out.println(names);

}

}

Output:

{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

c) Linked Hash Map:

 Maintains insertion order.

 Slower than Hash map.

 I can expect a faster iteration.

Example:

public class Fruit{

public static void main(String[ ] args){

LinkedHashMap<Sting,String> names =new LinkedHashMap<String,String>( );

names.put(“key1”,“cherry”);

names.put(“key2”,“apple”);

names.put(“key3”,“banana”);

names.put(“key4”,“kiwi”);

names.put(“key2”,“orange”);

System.out.println(names);

}

}

Output:

{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

d) TreeMap:

 Sorted Map.

 Like Tree set, we can construct a sort order with the constructor.

Example:

public class Fruit{

public static void main(String[ ]args){

TreeMap<Sting,String> names =new TreeMap<String,String>( );

names.put(“key1”,“cherry”);

names.put(“key2”,“banana”);

names.put(“key3”,“apple”);

names.put(“key4”,“kiwi”);

names.put(“key2”,“orange”);

System.out.println(names);

}

}

Output:

{key1=cherry, key2=banana, key3 =apple, key4=kiwi}

It is sorted in ascending order based on the key. Duplicate keys are not allowed.

Q #30) Explain the Priority Queue.

Answer: Queue Interface

Priority Queue: Linked list class has been enhanced to implement the queue interface.

Queues can be handled with a linked list. The purpose of a queue is “Priority-in, Priorityout”.

Hence elements are ordered either naturally or according to the comparator. The elements

ordering represents their relative priority.

Q #31) What is meant by Exception?

Answer: An Exception is a problem that can occur during the normal flow of execution. A

method can throw an exception when something wails at runtime. If that exception couldn’t

be handled, then the execution gets terminated before it completes the task.

If we handled the exception, then the normal flow gets continued. Exceptions are a subclass

of java.lang.Exception.

Example for handling Exception:

try{

//Risky codes are surrounded by this block

}catch(Exception e){

//Exceptions are caught in catch block

}

Q #32) What are the types of Exceptions?

Answer: There are two types of Exceptions. They are explained below in detail.

a) Checked Exception:

These exceptions are checked by the compiler at the time of compilation. Classes that

extend Throwable class except Runtime exception and Error are called checked Exception.

Checked Exceptions must either declare the exception using throws keyword (or)

surrounded by appropriate try/catch.

For Example, ClassNotFound Exception

b) Unchecked Exception:

These exceptions are not checked during the compile time by the compiler. The compiler

doesn’t force to handle these exceptions. It includes:

 Arithmetic Exception

 ArrayIndexOutOfBounds Exception

Q #33) What are the different ways to handle exceptions?

Answer: Two different ways to handle exception are explained below:

a) Using try/catch:

A risky code is surrounded by try block. If an exception occurs, then it is caught by the

catch block which is followed by the try block.

Example:

class Manipulation{

public static void main(String[] args){

add();

}

Public void add(){

try{

addition();

}catch(Exception e){

e.printStacktrace();

}

}

}

b) By declaring throws keyword:

At the end of the method, we can declare the exception using throws keyword.

Example:

class Manipulation{

public static void main(String[] args){

add();

}

public void add() throws Exception{

addition();

}

}

Q #34) What are the advantages of Exception handling?

Answer: The advantages are as follows:

 The normal flow of the execution won’t be terminated if exception got handled

 We can identify the problem by using catch declaration

Q #35) What are the Exception handling keywords in Java?

Answer: Enlisted below are the two Exception Handling Keywords:

a) try:

When a risky code is surrounded by a try block. An exception occurring in the try block is

caught by a catch block. Try can be followed either by catch (or) finally (or) both. But any

one of the blocks is mandatory.

b) catch:

This is followed by try block. Exceptions are caught here.

c) finally:

This is followed either by try block (or) catch block. This block gets executed regardless of

an exception. So generally clean up codes are provided here.

Q #36) Explain about Exception Propagation.

Answer: Exception is first thrown from the method which is at the top of the stack. If it doesn’t catch, then it pops up the method and moves to the previous method and so on until they are got.

This is called Exception propagation.

Example:

public class Manipulation{

public static void main(String[] args){

add();

}

public void add(){

addition();

}

From the above example, the stack looks like as shown below:

If an exception occurred in the addition() method is not caught, then it moves to the method add(). Then it is moved to the main() method and then it will stop the flow of execution. It is called Exception Propagation.

Q #37) What is the final keyword in Java?

Answer:

Final variable: Once a variable is declared as final, then the value of the variable could not be changed. It is like a constant.

Example:

final int = 12;

Final method: A final keyword in a method, couldn’t be overridden. If a method is marked

as a final, then it can’t be overridden by the subclass.

Final class: If a class is declared as final, then the class couldn’t be subclassed. No class

can extend the final class.

Q #38) What is a Thread?

Answer: In Java, the flow of execution is called Thread. Every java program has at least

one thread called the main thread, the main thread is created by JVM. The user can define

their own threads by extending the Thread class (or) by implementing the Runnable

interface. Threads are executed concurrently.

Example:

public static void main(String[] args){//main thread starts here

}

Q #39) How do you make a thread in Java?

Answer: There are two ways available to make a thread.

a) Extend Thread class: Extending a Thread class and override the run method. The

thread is available in java.lang.thread.

Example:

Public class Addition extends Thread {

public void run () {

}

}

The disadvantage of using a thread class is that we cannot extend any other classes because we have already extended the thread class. We can overload the run () method in our class.

b) Implement Runnable interface: Another way is by implementing the runnable

interface. For that, we should provide the implementation for the run () method which is

defined in the interface.

Example:

Public class Addition implements Runnable {

public void run () {

}

}

Q #40) Explain about join () method.

Answer: Join () method is used to join one thread with the end of the currently running

thread.

Example:

public static void main (String[] args){

Thread t = new Thread ();

t.start ();

t.join ();

}

From the above code, the main thread has started the execution. When it reaches the code

t.start() then ‘thread t’ starts the own stack for the execution. JVM switches between the

main thread and ‘thread t’.

Once it reaches the code t.join() then ‘thread t’ alone is executed and completes its task,

then only the main thread started the execution.

It is a non-static method. The Join () method has an overloaded version. So we can mention

the time duration in join () method also “.s”.

Q #41) What does the yield method of the Thread class do?

Answer: A yield () method moves the currently running thread to a runnable state and allows the other threads for execution. So that equal priority threads have a chance to run.It is a static method. It doesn’t release any lock.

Yield () method moves the thread back to the Runnable state only, and not the thread to

sleep (), wait () (or) block.

Example:

public static void main (String[] args){

Thread t = new Thread ();

t.start ();

}

public void run(){

Thread.yield();

}

}

Q #42) Explain about wait () method.

Answer: wait () method is used to make the thread to wait in the waiting pool. When the

wait () method is executed during a thread execution then immediately the thread gives up

the lock on the object and goes to the waiting pool. Wait () method tells the thread to wait

for a given amount of time.

Then the thread will wake up after notify () (or) notify all () method is called.

Wait() and the other above-mentioned methods do not give the lock on the object

immediately until the currently executing thread completes the synchronized code. It is mostly used in synchronization.

Example:

public static void main (String[] args){

Thread t = new Thread ();

t.start ();

Synchronized (t) {

Wait();

}

}

Q #43) Difference between notify() method and notifyAll() method in Java.

Answer: The differences between notify() method and notifyAll() method are

enlisted below:

notify() notifyAll()

This method is used to send a signal to wake up a single thread in the waiting pool.This method sends the signal to wake up all

the threads in a waiting spool.

Q #44) How to stop a thread in java? Explain about sleep () method in a thread?

Answer: We can stop a thread by using the following thread methods:

 Sleeping

 Waiting

 Blocked

Sleep: Sleep () method is used to sleep the currently executing thread for the given amount of time. Once the thread is wake up it can move to the runnable state. So sleep () method is used to delay the execution for some period.

It is a static method.

Example:

Thread. Sleep (2000)

So it delays the thread to sleep 2 milliseconds. Sleep () method throws an uninterrupted

exception, hence we need to surround the block with try/catch.

public class ExampleThread implements Runnable{

public static void main (String[] args){

Thread t = new Thread ();

t.start ();

}

public void run(){

try{

Thread.sleep(2000);

}catch(InterruptedException e){

}

}

Q #45) When to use the Runnable interface Vs Thread class in Java?

Answer: If we need our class to extend some other classes other than the thread then wecan go with the runnable interface because in java we can extend only one class.

If we are not going to extend any class then we can extend the thread class.

Q #46) Difference between start() and run() method of thread class.

Answer: Start() method creates a new thread and the code inside the run () method is executed in the new thread. If we directly called the run() method then a new thread is not

created and the currently executing thread will continue to execute the run() method.

Q #47) What is Multi-threading?

Answer: Multiple threads are executed simultaneously. Each thread starts its own stack based on the flow (or) priority of the threads.

Example Program:

public class MultipleThreads implements Runnable

{

public static void main (String[] args){//Main thread starts here

Runnable r = new runnable ();

Thread t=new thread ();

t.start ();//User thread starts here

Addition add=new addition ();

}

public void run(){

go();

}//User thread ends here

}

On the 1st line execution, JVM calls the main method and the main thread stack looks as shown below.

Once the execution reaches, t.start () line then a new thread is created and the new stack for the thread is also created. Now JVM switches to the new thread and the main thread are back to the runnable state.

The two stacks look as shown below.

Now, the user thread executed the code inside the run() method.

Once the run() method has completed, then JVM switches back to the main thread and the

user thread has completed the task and the stack was disappeared.

JVM switches between each thread until both the threads are completed. This is called Multithreading.

Q #48) Explain the thread life cycle in Java.

Answer: Thread has the following states:

 New

 Runnable

 Running

 Non-runnable (Blocked)

 Terminated

 New: In New state, a Thread instance has been created but start () method is not

yet invoked. Now the thread is not considered alive.

 Runnable: The Thread is in the runnable state after the invocation of the start ()

method, but before the run () method is invoked. But a thread can also return to the

runnable state from waiting/sleeping. In this state, the thread is considered alive.

 Running: The thread is in running state after it calls the run () method. Now the

thread begins the execution.

 Non-Runnable(Blocked): The thread is alive but it is not eligible to run. It is not in

the runnable state but also, it will return to the runnable state after some time.

Example: wait, sleep, block.

 Terminated: Once the run method is completed then it is terminated. Now the

thread is not alive.

Q #49) What is Synchronization?

Answer: Synchronization makes only one thread to access a block of code at a time. If multiple threads accesses the block of code, then there is a chance for inaccurate results at

the end. To avoid this issue, we can provide synchronization for the sensitive block of codes.

The synchronized keyword means that a thread needs a key in order to access the synchronized code.

Locks are per objects. Every Java object has a lock. A lock has only one key. A thread can

access a synchronized method only if the thread can get the key to the objects to lock.

For this, we use the “Synchronized” keyword.

Example:

public class ExampleThread implements Runnable{

public static void main (String[] args){

Thread t = new Thread ();

t.start ();

}

public void run(){

synchronized(object){

{

}

}

Q #50) What is the disadvantage of Synchronization?

Ans: Synchronization is not recommended to implement all the methods. Because if one thread accesses the synchronized code then the next thread should have to wait. So it makes a slow performance on the other end.

Q #51) What is meant by Serialization?

Answer: Converting a file into a byte stream is known as Serialization. The objects in the file are converted to the bytes for security purposes. For this, we need to implement a java.io.Serializable interface. It has no method to define.

Variables that are marked as transient will not be a part of the serialization. So we can skip the serialization for the variables in the file by using a transient keyword.

Q #52) What is the purpose of a transient variable?

Answer: Transient variables are not part of the serialization process. During deserialization,

the values of the transient variables are set to the default value. It is not used with static variables.

Example:

transient int numbers;

Q #53) Which methods are used during the Serialization and Deserialization

process?

Answer: ObjectOutputStream and ObjectInputStream classes are higher level java.io.

package. We will use them with lower level classes FileOutputStream and FileInputStream.

ObjectOutputStream.writeObject —->Serialize the object and write the serialized object to a

file.

ObjectInputStream.readObject —> Reads the file and deserializes the object.

To be serialized, an object must implement the serializable interface. If superclass implements Serializable, then the subclass will automatically be serializable.

Q #54) What is the purpose of a Volatile Variable?

Answer: Volatile variable values are always read from the main memory and not from thread's cache memory. This is used mainly during synchronization. It is applicable only for

variables.

Example:volatile int number;

Q #55) Difference between Serialization and Deserialization in Java.

Answer: These are the difference between serialization and deserialization in java: Serialization Deserialization Serialization is the process which is used

to convert the objects into byte stream Deserialization is the opposite process of

serialization where we can get the objects back from the byte stream.

An object is serialized by writing it an

ObjectOutputStream.

An object is deserialized by reading it from an ObjectInputStream.

Q #56) What is SerialVersionUID?

Answer: Whenever an object is Serialized, the object is stamped with a version ID number for the object class. This ID is called the SerialVersionUID. This is used during deserialization to verify that the sender and receiver that are compatible with the

Serialization.