**Core\_Java\_Interview\_Questions**

**Q1) What is OOP’S**

1. Object oriented programming (oops) is methodology that simplifies software development and maintenance by providing some rules.

Basic Components of OOP’s:

* Class
* Object
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Q2) In how many ways we can create object in Java**

A) We can create object in five ways.

1. By New keyword
2. By New Instance() method
3. By clone() method
4. By deserialization
5. By factory method

**Q3) Can we have two methods with the same name in a class**

A) Yes, but we should change return type and arguments count

**Q4) What is static keyword, static variable, static method and static block in java**

A) **Static Keyword:** The static keyword in java is used for memory management mainly.

**Static Variable:** The static variable can be used to refer the common property of all objects.

The static variable gets memory only once in a class area at the time of class loading

**Static Method:** The static method belongs to the class rather than object of class. The static method can be invoked without creating the instance of class.

The static method can access static data members and can change the value of it.

**Restrictions for static method:** The static method cannot use the non-static data members or call non static data members directly.

This and Super keywords cannot be used in static context.

**Static Block:** It is used to initialize the static data members. It is executed before main method at the time of class loading

**Q5) Can we have two methods with the same name in a class**

**A)** Yes, we can have but we should change return type and arguments count.

**Q6) What is the difference between abstract class and interface**

**A)** If the class having at least one abstract method then the class should be created as an abstract class

An abstract class can contain both abstract and concurrent methods.

Interface contains only abstract methods.

Abstract class can contain four types static, non-static, final and non-final variable.

Interface contains only static and final variable.

To declare abstract class, we should use the abstract keyword.

To declare interface, we should use the interface keyword.

The keyword “extends” is used to extend the abstract class

The keyword “implements” is used to implements the interface

Abstract class doesn’t support the multiple inheritance.

Interface supports the multiple inheritance.

**Q7) If a method throws null pointer exception in super class, can we override it with a method throws runtime exception.**

**A)** Yes, we can override that method because both exceptions we mentioned here are Runtime Exceptions (Actually NullPointerException is a sub class of Runtime Exception). Which are unchecked exceptions this means that they are not checked at compile time.

**Q8) Difference between .equals() method and (==) operator**

**A)** Both equals() method and == operator used to compare the two objects in java.

== operator is used to check the memory location or address of the two objects are same or not.

.equals() method is used to check the content comparison, to check both the objects content same or not.

**Q9) What type of garbage collector are provided and how we can choose any of the garbage collector.**

**A)** In java garbage collection is a mechanism that provides automatic memory management, it is done by JVM. It need not to handle object allocation and de-allocation by the programmer.

There are 4 types of garbage collectors.

* Serial garbage collector
* Parallel garbage collector
* Concurrent mark sweep garbage collector
* Garbage first(G1) garbage collector

-XX:+UseSerialGC – Serial garbage collector

-XX:+UseParallelGC – Parallel garbage collector

-XX:+UseConcurrentmarksweepGC – CMS collector

-XX:+UseG1GC – G1 garbage collector

We can call garbage collector like below

System.gc()

**Q10) What is big integer**

**A)** The big integer is nothing but a Integer wrapper class. A value of primitive type int in an object.

**Q11) What is the clonable interface**

**A)** A clonable interface is used to create the exact copy of object. It exists in java.lang.package

A class must implement the clonable interface if we want to create the clone of the object.

If we don’t implement clonable interface, clone () method generates CloneNotSupportedException.

A clonable interface is a marker interface.

**Q12) What is marker interface**

**A)** A marker interface is an empty interface (no files or methods). If we want to assign additional information to the object then we will implement from the marker interface.

Serializable and Clonable are the marker interfaces

**Q13) Why core java is not 100% object oriented**

**A)** Java is not a “pure” object-oriented language, because it uses primitive data types such as (int, long, float, char…… etc)

The developers of java could have made these primitive data types as object (wrapper classes).

**Q14) Does java supports multiple inheritance**

**A)** Java does not support multiple inheritance but we can achieve multiple inheritance by using interfaces in java.

**Q15) Can we override static methods**

**A)** No, we can’t override static methods because method override is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time, So we cannot override static methods

**Q16) String Reverse program**

**A)** String str=”srinu”;

For(int i=0;i<str.length();i++){

Char ch=str.charAt();

String nstr=ch+nstr;

}

s.o.p(“reversed:”+nstr)

**Q17) Object class methods**

**A)** There are 11 methods are there in object class

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Hashcode() | Returns the hashcode value of the method |
| Finalize() throws throwable | Called by the garbage collector |
| .equals() | Uses for comparing two objects |
| Clone() | Create a exact copy of the object |
| toString() | Returns the string representation of the object |
| Getclass() | Returns present class reference |
| Notify()  notifyAll()  wait()  wait(long timeout)  wait(long timeout,int nano’s) | All these methods used in java multithreading, which plays a critical role in synchronization |

**Q18) Why string is immutable**

**A)** The string is immutable in java because of the security, synchronization and concurrency. Suppose there are 5 reference variables, all refers to one object. If one reference variable changes the value of the object, it will be affected to all the reference variables. That’s why String is objects are immutable.

**Q19) What is the difference between heap and stack memory in Java.**

**A) Stack:** Java stack memory is used to execution of a thread, It contain method specification values. That are short lived and reference the other object in the heap that is getting referred from the method. When the method execution is done the stack memory will get destroy.

**Heap:** Java heap memory is used by java runtime to allocate memory to objects and JRE classes. We create an object’s; it’s always created in the heap memory.

**Q20) Design patterns in java**

**A)** There are 6 design patterns I java

1. **Singleton design pattern :** Suppose if we want to create class for which only a single instance or object should be created and that single object can be used by all other classes is called singleton design pattern.

Singleton class must provide a global access point to the instance of the class.

Singleton pattern is used for logging, drivers object, caching and thread pool.

1. **Prototype design pattern:** Prototype design pattern is cloning of an existing object instead of creating new one and can also be customized as per the requirement.
2. **Factory method pattern:** Factory design pattern is used where we have a super class with multiple sub-classes and based on input we need to return one of the sub-class.
3. **AbstractFactory pattern:** Abstract factory pattern is similar to factory pattern and it is a factory of factories.

We have factory class that returns the different sub-classes based on the input provided and factory class uses if-else or switch statement to achieve this.

1. **Object pool design pattern:** It is to reuse object that are expensive to create.
2. **Builder Design pattern:** This pattern was introduced to solve some of the problems with factory and abstract design patterns when the object contains a lot of attributes.

Builder is with large number of optional parameters and inconsistence state by providing a way to build the object.

**Q21) What is the use of Constructor**

**A)** A constructor is similar to method have no return type and it is invoked at the time of creating an object of the class. It is generally used to initialize the instance variable of a class. The constructors have same name as their class name.

There are two types of constructors, parameterized constructor and no-arg constructors. The main purpose of a constructor is to initialize the object of a class.

**Q22) What is deadlock**

1. When ‘n’ threads are in waiting state in such a way that thread-1 is waiting for some resource which is held by thread-2 and thread-2 is waiting for some resource which is held by thread-3 and so on thread-n is waiting for some resource which is held by thread-1, this situation when all the threads are in waiting state and no thread release the lock because it is also in waiting state is known as deadlock.

**Q23) How can we handle Exceptions**

**A)** In two ways we can handle the exception

* **Try-catch blocks:** In this we can write the rescue code in try block and exception in catch block
* **Throws keyword:** Throws keyword is used to declare an exception. It gives the programmer that there are many occur an exception so it is better for the programmer to provide exception handling code.

**Q24) What is the difference between Thread class and Runnable interface**

**A)**

|  |  |
| --- | --- |
| **Thread** | **Runnable** |
| It is a class | It is a interface |
| It can be used to create thread | It can also used to create thread |
| It has multiple methods such as ‘run’, ’start’ and ‘sleep’ methods | It has only single abstract method run() |
| Since multiple inheritance is not allowed in java, hence after a class extends the thread class | When a class implements runnable interface the class can extended to other classes |
| Every thread creates a unique object and associate with it | Multiple threads can share the same object |

**Q25) Can we call start() method multiple times**

**A)** No, after starting a thread, it can never be started again. If you do so illegelThreadstate exception is thrown. In such case thread will once but for second time it will throw exception.

However, the run method can be called multiple times as it will be treated as a normal function call.

**Q26) What is the purpose of wait(), notify() and notifyAll() methods**

**A)** Wait(): The wait() method causes the current thread to wait until another thread invokes the notify() or notifyAll() methods for that object.

Notify(): The notify() method wakes up a single thread that is waiting on that object monitor.

NotifyAll(): The notifyAll() method wakes up all threads that are waiting on that object monitor

**Q27) How we will initialize the values in immutable class**

**A)** We can initialize all the values via parameterized constructor that performs the deep copy.

Performing cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

**Q28) How to create a immutable class**

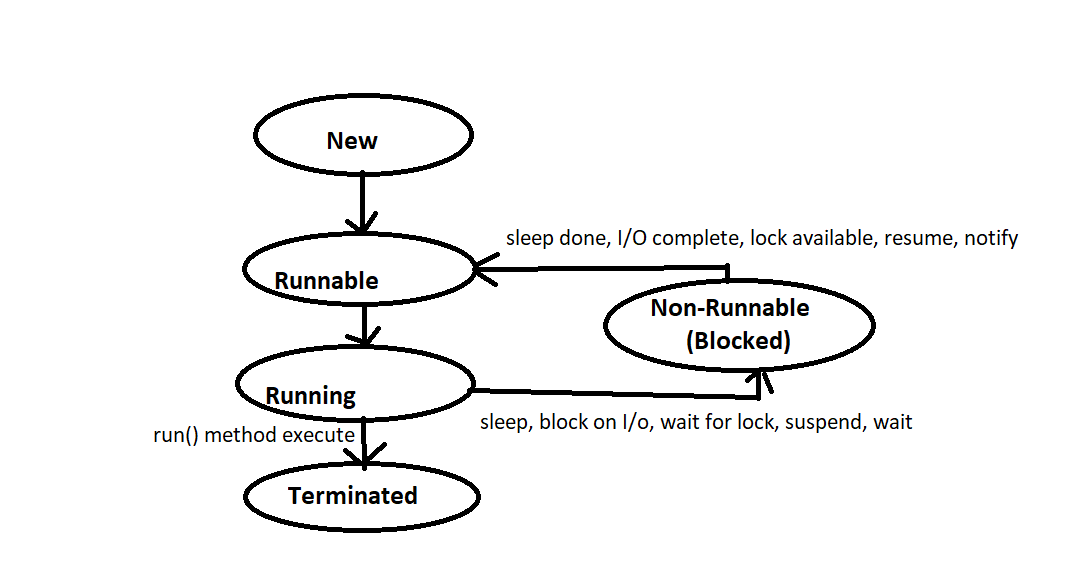
**A)** Declare the class as final so it can’t be extended

Make all the fields as final and private, So that direct access is not allowed

Make all mutable fields final so that their values can be assigned only once.

Use the getter method for all the variables in it.

**Q29) Life cycle of a Thread**

**A)**  

* **New:** A Thread is in new when it gets CPU time and it’s begins it’s life cycle in the new state.
* **Runnable:** A thread in this state is considered to be either running or ready for execution but it’s waiting for resource allocation.
* **Suspended:** A thread is in the suspended state when it is temporarily inactive or under execution
* **Blocked:** A thread is in the blocked state then it is waiting for resource
* **Waiting:** Sometimes a thread transitions to the waiting state while the thread waits for another thread to perform a task.
* **Terminates:** A runnable thread enters the termination state when it completes it’s task or otherwise terminates

**Q30) What is Join() method in thread**

**A)** Join() method allows one thread to wait until another thread execution is completes it’s execution. In other words, it will wait for other thread to die. It has a void type

Join()

Join(long milli’s)

Join(long milli’s, int nano’s)

**Q31) What is Yield() method**

**A)** A yield() method can stop the currently executing thread and will give chance to other waiting thread of the same priority.

Suppose there are three threads t1, t2 and t3. Thread t1 gets the processor and starts it’s execution and thread t2 and t3 are in runnable state.

The completion time for thread t1 is 5 hours and the completion time for t2 is 5 minutes. Since t1 will complete it’s execution after 5 hours, t2 has to wait for 5 hours to just finish 5 minuets job in such scenarios where one thread is taking too much time to complete it’s execution.

**Q32) What is singleton class**

**A)** Singleton class can have only one object (an instance of the class) at a time. After the first time, if we are trying to instantiate the singleton class, the new variables also point to the first instance created.

So whatever modifications we do on any variable inside the class through any instance, affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.

The primary purpose of a singleton class is to restrict the limit of the number of object creations.

For example database connection. Memory space wastage does not occur the use of the singleton class, because it restricts the instance creation. We can use the single object repeatedly as per requirement.

**Q33) How to create singleton class**

**A)** We need to create data members as private.

We need to create constructor as private, so that no other class can create instance for these class.

And we should not create any setter methods.

The class also to be final.

Write a static method, that has the return type object of the singleton class, here the concept of lazy initialization is used to write this static method.

**Q34) What is the serialization and De-Serialization**

**A)** Serialization is a mechanism of converting the state of object into byte stream.

Deserialization is the reverse process, where the byte stream is used to recreate the actual java object in memory.

The byte stream created is platform independent so, the object serialization on one platform can be done, deserialization on a different platform.

T o make a java object serialization we can import the java.io.serialization package.

The object output stream class contains writeobject() method for serialization an object.

The object input stream class contains readobject() method for deserialization an object.

**Q35) What is the executor framework in multithreading environment**

**A)** Java has strong support for multithreading programming, before java-5 the support was in the form of calling native constructors itself in the application layer. To overcome these issue java-5 has introduced executor framework.

The executor framework contains a bunch of components that are used to efficiently manage multiple threads. Which is used to run the runnable objects without creating new thread every time and also mostly re-using the already created threads.

There are mainly 4 types of executors available

* **Single Thread Executor:** This executor has only one thread and is used to execute task in a sequential manner.
* **Fixedpool Executor:** This is a pool of a fixed number of threads
* **Cached Thread Executor:** This is mainly used when there are lots of short lived parallel task on the line waiting to be executed.
* **Scheduled Executor:** This executor is used when you have a task that needs when you have a task that needs to be run at regular intervals.

**Methods of java executor service:**

i await termination() ii ListinvokeAll()

iii invokeAny() iv Boolean isShutDwon()

v Boolean isTerminate() vi void shutdown

vii List shoutdownNow()

**Q36) How do we remove duplicates in array**

**A)**

public class Main {

    public static int removeduplicates(int a[], int n)

    {

        if (n == 0 || n == 1) {

            return n;

        }

        // creating another array for only storing

        // the unique elements

        int[] temp = new int[n];

        int j = 0;

        for (int i = 0; i < n - 1; i++) {

            if (a[i] != a[i + 1]) {

                temp[j++] = a[i];

            }

        }

        temp[j++] = a[n - 1];

        // Changing the original array

        for (int i = 0; i < j; i++) {

            a[i] = temp[i];

        }

        return j;

    }

    public static void main(String[] args)

    {

        int a[] = { 1, 1, 2, 2, 2 };

        int n = a.length;

        n = removeduplicates(a, n);

        // Printing The array elements

        for (int i = 0; i < n; i++)

            System.out.print(a[i] + " ");

    }

}

**Q37) Reverse of single linkedList**

**A)**

class LinkedList {  
 static Node *head*;  
 static class Node {  
 int data;  
 Node next;  
 Node(int d)  
 {  
 data = d;  
 next = null;  
 }  
 }  
 Node reverse(Node node)  
 {  
 Node prev = null;  
 Node current = node;  
 Node next = null;  
 while (current != null) {  
 next = current.next;  
 current.next = prev;  
 prev = current;  
 current = next;  
 }  
 node = prev;  
 return node;  
 }  
 void printList(Node node)  
 {  
 while (node != null) {  
 System.*out*.print(node.data + " ");  
 node = node.next;  
 }  
 }  
 public static void main(String[] args)  
 {  
 LinkedList list = new LinkedList();  
 list.*head* = new Node(10);  
 list.*head*.next = new Node(20);  
 list.*head*.next.next = new Node(30);  
 list.*head*.next.next.next = new Node(40);  
 System.*out*.println("old Linked list");  
 list.printList(*head*);  
 *head* = list.reverse(*head*);  
 System.*out*.println("");  
 System.*out*.println("new linked list ");  
 list.printList(*head*);  
 }  
}

**Q38) How to use lambda expression in functional interface**

**A)** We can use Lambda expression in functional interface by using three components

i.) Arguments List: it can be empty or non-empty

ii.) Array-token: it is used to link arguments list and body of the expression

iii.) Body: it contains expression and statements for lambda expression

Ex-1:

Interface Sayable{

Public String say(String name);

}

Public class LambdaExpression{

Public static void main(String[] args){

Sayable s=(name)->{

Return “Hello”+name;

}

System.out.println(s.say(“Srinu”))

Ex-2:

Interface addable{

Public add(int a,int b);

}

Public class LambdaExpression{

Public static void main(String[] args){

Add ad=(int a, int b)->(a+b);

System.out.println(ad.add(10,20))

}

}

**Q39) What is null pointer exception and how to handle it before java-8**

**A)** Null pointer exception is a runtime condition where we try to access or modify an object which has not been initialized yet. It means that the object’s reference variable is not pointing anywhere and refers to nothing or “null”.

**Q40) What is the use of This and Super keywords**

**A) This:** This is a reference variable that refers to current class objects.

* This can be used to refer current class instance variable
* This can be used to invoke current class method
* This() can be used to invoke the current class constructor.
* This can be used to pass as an arguments in the method call
* This can be passed as an arguments in the constructor call.
* This can be used to return the current class instance from the method.

**Super:** The super keyword in java is a reference variable which is used to refer immediate parent class object.

* Super can be used to refer immediate parent class variables.
* Super can be used to invoke immediate parent class method
* Super() can be used to invoke immediate parent class constructor

**Q41) What is aggregation in java**

**A)** If a class have an another entity class reference, it is known as aggregation. Aggregation represents the HAS-A relationship

Ex: class Employee{

Int id;

String name;

Address address;

}

**Q42) What is final keyword and its members**

**A)** The final keyword in java is used to restrict the user. The final keyword can be used in many contexts.

* Variable
* Method
* Class

Final Variable: If you make any variable as final, you cannot change the value of final variable (it will be constant)

Final Method: If we make any method as final, we cannot override it.

Final Class: If we make any class as final, we cannot extend it.

**Q44) What is checked and unchecked exceptions**

**A)** Checked exceptions are checked at compile time.

Ex: IOExceptions, SQL Exceptions etc..

The classes that extended throwable class except RuntimeException and Error

Unchecked exceptions are not checked at compile time rather than that checked at runtime.

Ex: Athematic Exceptions, NullPointer Exception, ArrayOutOfBound Exception etc..

**Q45) What is the difference between Throw and Throws keywords**

**A)** Throw: The throw keyword is used to explicitly throw an exception. We can through either checked exception or unchecked exception by using throw keyword. It is mainly used to throw custom exception.

Throws: Throws keyword is used to declare an exception. It is better for the programmer to provide the exception handing code.

|  |  |
| --- | --- |
| Throw | Throws |
| Java Throw keyword is used to explicitly throw an exception | Java throws keyword is used to declare an exception |
| Throw is followed by a instance | Throws is followed by a class |
| Throw is used within the method | Throws is used with the method signature |
| You can’t throw multiple exceptions at a time | You can declare multiple exception at a time |

**Q46) What is difference between Final, Finally and Finalize**

**A) Final:** Final is used to apply restrictions on class, method and variable. Final class can’t be inherited, final method can’t be overridden and final variable can’t be changed

**Finally:** Finally, is used to place important code, it will be executed whether exception is handled or not.

**Finalize:** Finalize is used to perform clean-up processing just before object is garbage collector

**Q47) What is Anonymous Inner class**

1. A class that have no name is known as anonymous inner class in java.

It should be used if you have to override method of a class or interface.

Ex: abstract class person{

Abstract void eat();

}

Class TestAnonymousInner{

PSVM(String args[]){

Person p=new person(){

Void eat(){

s.o.p(“nice fruit”);

}

};

p.eat();

}

}

**Q48) What is the difference between wait and sleep methods**

**A)**

|  |  |
| --- | --- |
| **Wait()** | **Sleep()** |
| The wait method is defined in object class | The sleep method defined in thread method. |
| The wait method releases the lock | The sleep method doesn’t release the lock |
| Should be notified by notify() or notifyAll() methods | After specified amount of time, sleep is completed. |
| Wait is not static method | Sleep is a static method |
| We will call the wait method only from the synchronized context | We will call the sleep method from outside the synchronized context |

**Q49) can we have try block with catch?**

**A)** Yes, It is possible to have a try block without a catch block by using a final block.

As we know, a final block will always execute even there is an exception occurred in a try block, except System.exit() it will execute always.

**Q50) How we will consider the string is palindrome or not**

1. I can reverse the given string by using Stringbuffer then I will compare the both the strings by using .equals method, if both the strings same then it is palindrome or else it is not palindrome.