# Title : Core Java Interview Question and Answers.

## 1) What are static blocks and static initializers in Java ?

- static blocks or static initializers are used to initialize static fields in java.
- We declare static blocks when we want to initialize static fields in our class.
- static blocks get executed exactly once when the class is loaded .
- static blocks are executed even before the constructors are executed.
- static blocks will execute even before the main method as well.
- We can have any number of static blocks in one single class.
- Static block scope is always a class scope and we can execute outside of the class and outside of the package as well.

## 2) What is constructor chaining or How to call one constructor from the other constructor ?

- Within the same class if we want to call one constructor from another we use this() method.
- Based on the number of parameters we pass appropriately this() method is called.
- Restrictions for using this method :
  1) this() must be the first statement in the constructor.
  2) We cannot use two this() methods in the constructor.

- If we want to call a superclass constructor we can call super() method.
- Restrictions for using super method :
  1) super() must be the first statement in the child class constructor.

2) We cannot use two super() methods in the constructor.
3) We cannot use super() and this() in the same constructor.

# 3) What is method overriding in java ?

- **Whenever we are not satisfied or happy with parent class functionalities** then we can go for the child class with different functionalities with the same signature method called method overriding.
- If we have methods with the **same signature(method name + arguments)** in super class and subclass then we say subclass method is overridden by superclass.
- When to use overriding in java, If we want the same method with different behavior in subclass then we go for overriding.
- When we call an overridden method with a subclass reference subclass method is called hiding the superclass method.
- Method Overriding rules
  1) Method signatures should be the same in parent and child classes.
  2) Method return type **should be the same**
     - **if there are primitive data types** From parent to child.
     - **if there are Object data types**, no need to be the same after 1.5 version so we can use covariant return types also.
     - covariant return types means
     - As the ear hit eardrums ``overriding" we quickly get to know that it can be done either by virtue of different data types or arguments passed to a function that a programmer learned initially while learning polymorphism in java.
     - Before JDK 5.0, it was not possible to override a method by changing the return type.

- When we override a parent class method, the name, argument types, and return type of the overriding method in child class has to be exactly the same as that of the parent class method.
- The overriding method was said to be invariant with respect to return type.
- **Example** :
- Parent class ---> Child class
- Number --> Integer, Double, Float etc...
- Object --> String, StringBuffer

3) Method access modifiers scope should not be reduced from parent to child. (default < protected < public)
   - If the parent class method **is defaul**t, the child class method can be **default, protected and public**.
   - If the parent class method is **protected** , the child class method can be **protected and public**.
   - If the parent class method is **public**, the child class method can be **public**.
4) If parent class methods are private **we can not override the private methods** in child class methods.There are no errors too.
5) If parent class methods are static we can not say method overriding but can say that is called method hiding.
6) If the Parent class method throws any exception, the child class method does not need to throw any exception but if the child class throws any exception parent class method should throw the same exception or its parent.

Ex : run() method in multithreading concept is the best example.

## 4) What is method overloading in java ?

- A class having two or more methods with the same name but with different arguments then we say that those methods are overloaded.
- Static polymorphism is achieved in java using method overloading.
- Method overloading is used when we want the methods to **perform similar tasks** but with different inputs or values.
- When an overloaded method is invoked java first checks the method name, and the number of arguments ,type of arguments; based on this, the compiler executes this method.
- Compiler decides which method to call at compile time.
- By overloading static polymorphism or static binding can be achieved in java.
  Note : Return type is not part of method signature. we may have methods with different return types but return type alone is not sufficient to call a method in java.

Ex: println() method in PrintStream class is the best example.

## 5) What is the super keyword in java ?

- Variables and methods of super class can be overridden in subclass .
- In case of overriding, a subclass object calls its own variables and methods.
- Subclass cannot access the variables and methods of superclass because the overridden variables or methods hide the methods and variables of superclass.
- But still java provides a way to access super class members even if its members are overridden.
- Super is used to access superclass variables, methods, constructors.
  Super can be used in two forms :
  1) First form is for calling super class constructors.
  2) Second one is to call super class variables,methods.
- Super if present must be the first statement.

## 6) Difference between method overloading and method overriding in java ?

| Method Overloading | Method Overriding |
|---|---|
| - Method Overloading occurs within the same class | - Method Overriding occurs between two classes superclass and subclass |
| - Since it involves only one class inheritance is not involved. | - Since method overriding occurs between superclass and subclass inheritance is involved. |
| - In overloading, the return type need not be the same. | - In overriding the return type must be the same up to 1.8 and after that we can use covariant return types also. |
| - Parameters must be different when we do overloading, count of the arguments no need to be the same. | - Parameters must be the same. |
| - Static polymorphism can be achieved using method overloading | - Dynamic polymorphism can be achieved using method overriding. |
| - In overloading one method can't hide the another | - The overriding subclass method hides that of the superclass method. |

## 7) Difference between abstract class and interface ?

| Interface | Abstract class |
|---|---|

| | |
|---|---|
| - If we don't know anything about implementation and just have requirement specifications then we should go for interfaces. | - If we are talking about implementation but not completely (partially implementation ) then we should go for abstract class. |
| - interface contains only abstract methods up to 1.8, From 1.8 onwards interface can have default and static methods. And 1.9 we have private methods . | - abstract class can contain abstract methods, concrete methods or both |
| - Access Specifiers for methods in interface must be public up to 1.8, from 1.8 we can have static also and 1.9 we have private methods. | - Except private we can have any access specifier for methods in abstract class. |
| - Variables defined must be public , static and final | - Except private variables can have any access specifiers. |
| - Multiple Inheritance in java is implemented using interface | - We cannot achieve multiple inheritance using abstract classes. |
| - To implement an interface we use implements keyword | - To implement abstraction we use extends keyword |
| - For interface variables compulsory we should perform initialization at the time of declaration only otherwise we will get compile time errors. | - For interface variables compulsory we should perform initialization at the time of declaration only otherwise we will get compile time errors. |

| | |
|---|---|
| - Inside the interface we can't declare instance and static blocks. | - Inside abstract class can declare instance and static blocks. |
| - Inside the interface we can't declare constructors. | - inside abstract we can declare constructors |

# 8) Why is java platform independent?

- The most unique feature of java is platform independent.
- In any programming language source code is compiled into executable code. This cannot be run across all platforms.
- When javac compiles a java program it generates an executable file called .class file.
- class file contains byte codes. Byte codes are interpreted only by JVM's .
- Since these JVM's are made available across all platforms by Sun Microsystems, we can execute this byte code in any platform.
- Byte code generated in the Windows environment can also be executed in the Linux environment.
- This makes the java platform independent.

# 9) What is the difference between c++ and Java ?

| Java | c++ |
|---|---|
| - Java is platform independent. | - C++ is platform dependent. |
| - There are no pointers in java. | - There are pointers in C++. |
| - There is no operator overloading in java. | - C ++ has operator overloading. |

| | |
|---|---|
| - There is garbage collection in java. | - There is no garbage collection. |
| - Supports multithreading. | - Does Not support multithreading. |
| - There are no templates in java. | - There are templates in java. |

## 10) What is a JIT(just-in-time) compiler ?

- JIT compiler stands for Just in time compiler.
- JIT compiler compiles byte code into executable code .
- JIT is a part of JVM.
- JIT cannot convert a complete java program in to executable code it converts as and when it is needed during execution.

## 11) What is bytecode in java ?

- When a javac compiler compiles a class it generates a .class file.
- This .class file contains a set of instructions called byte code.
- Byte code is a machine independent language and contains a set of instructions which are to be executed only by JVM. JVM can understand this byte code.

## 12) Difference between this() and super() in java ?

- this() is used to access one constructor from another within the same class while super() is used to access superclass constructors.
- Either this() or super() exists; it must be the first statement in the constructor.

# 13) What is a class ?

- Classes are fundamental or basic units in Object Oriented Programming .
- A class is a kind of blueprint or template or plan or design or Overview or architecture for objects.
- Class defines variables, methods.
- A class tells what type of objects we are creating.
- For example, taking Department class tells us we can create department type objects.
- We can create any number of department objects.
- All programming constructs in java reside in class.
- When JVM starts running it first looks for the class when we compile.
- Every Java application must have at least one class and one main method.
- Class starts with the class keyword.
- A class definition must be saved in a class file that has the same as class name when the class is public..
- File name must end with .java extension.

```
public class HelloJava{
        public static void main(String[] args) {
                System.out.println("My First class");
            }
 }
```

- if we see the above class when we compile JVM loads the HelloJava and generates a .class file(Hello.class).
- When we run the program we are running the class and then executes the main method.

# 14) What is an Object ?

- An Object is an instance or Example of class.
- A class defines a type of object.
- Each object belongs to some class.
- Every object contains state and behavior.
- State is determined by value of attributes and behavior is called method.
- Objects are also called as an instance.
- To instantiate the class we declare it with the class type.

```
public class HelloJava{
        public static void main(String[] args) {
            HelloJava h = new HelloJava();// Creating an Object.
             System.out.println("My First class");
        }
}
```

# 15) What is the method in java ?

- It contains the executable body that can be applied to the specific object of the class.
- Method includes method name, parameters or arguments and return type and a body of executable code.
  ex : public float add(int a, int b, int c) methods can have multiple arguments.
  Separate with commas when we have multiple arguments.
  Syntax: returntype methodName(ArgumentList ){}

# 16) What is encapsulation ?

- The process of wrapping or binding or bundling or putting up data with related functionalities into a single unit class and keeping data safe from misuse is called encapsulation.
- Through encapsulation we can hide and protect the data stored in java objects.
- Java supports encapsulation through access control.
- There are four access control modifiers in java: public , private ,protected and default level.

- For example, take a car class. In a car we have many parts which are not required for the driver to know what all it consists of. He is required to know only about how to start and stop the car.
- So we can expose what all are required and hide the rest by using encapsulation.

- Another Best example for Encapsulation is Java API (keeping one class or one package, Java API is giving n number of classes and n number packages, Creating each class or each package is the concept of Encapsulation Mechanism.)

## - Why Encapsulation ?

- To Achieve Security or Modularity..

## - Where are we using Encapsulation ?

- Everywhere in java, Best example we can explain java API.

# 17) Why is the main() method public, static and void in java ?

- public : “public” is an access specifier which can be used outside the class. When the main method is declared public it means it can be used outside class.

- static : To call a method we require an object.
  Sometimes it may be required to call a method without the help of an object. Then we declare that method as static.JVM calls the main() method without creating objects by declaring keyword static.

- void : void return type is used when a method doesn't return any value .
  main() method doesn't return any value, so main() is declared as void.

## 18) Explain about the main() method in java ?

- Main() method is the starting point of execution for all java applications.
- public static void main(String[] args) {}
- String args[] are an array of string objects we need to pass from command line arguments.
- Every Java application must have at least one main method.
- JVM will call the main method to execute.

## 19) What is constructor in java ?

- A constructor is a special method used to initialize objects in java.
- We use constructors to initialize all variables in the class when an object is created.
- As and when an object is created it is initialized automatically with the help of a constructor in java.
- We have three types of constructors
  Default Constructor, no arg constructor and Parameterized Constructor
  Signature : public classname() { } when we created an Object and if there is no arg constructor in a class.
  Signature : public classname() { } explicitly if we have a class.
  Signature : public classname(parameters list) { }.

## 20) What is the difference between length and length() method in java ?

length() : In the String class we have a length() method which is used to return the number of characters in a string.
Ex : String str = “Hello World”;
System.out.println(str.length());
Str.length() will return 11 characters including space.

length : We have a length instance variable in arrays which will return the number of values or objects in the array.
For example : String days[]={” Sun”,”Mon”,”Tue”,”Wed”,”Thu”,”Fri”,”Sat”};
Will return 7 since the number of values in the days array is 7.

## 21) What is ASCII Code?

- ASCII stands for American Standard code for Information Interchange.
- ASCII character range is 0 to 255.
- We can’t add more characters to the ASCII Character set.
- The ASCII character set supports only English.
- That is the reason, if we see C language we can write C language only in English we can’t write in other languages because it uses ASCII code.

## 22) What is Unicode ?

- Unicode is a character set developed by the Unicode Consortium.
- To support all languages in the world Java supports Unicode values.

- Unicode characters are represented by 16 bits and its character range is 0-65,535.
- Java uses ASCII code for all input elements except for Strings,identifiers, and comments.
- If we want to use telugu we can use telugu characters for identifiers.
- We can enter comments in telugu also.

## 23) Difference between Character Constant and String Constant in Java ?

- Character constant is enclosed in single quotes.
- String constants are enclosed in double quotes.
- Character constants are single digit or character.
- String Constants are collections of characters.
- Ex for char :'2', 'A', 65, '\u0010'
- Ex for String : "Hello World"

## 24) What are constants and how to create constants in java?

- Constants are fixed values whose values cannot be changed during the execution of a program.
- We create constants in java using the final keyword.
- Ex : final int number =10;
- final String str="java-interview –question";

## 25) Difference between '>>' and '>>>' operators in java?

- >> is a right shift operator that shifts all of the bits in a value to the right to a specified number of times.
- int a =15; a= a >> 3;
- The above line of code moves 15 three characters right.
- >>> is an unsigned shift operator used to shift right.
- The places which were vacated by shift are filled with zeroes.

## 26) Explain Java Coding Standards for classes or Java coding conventions for classes?

- Sun has created Java Coding standards or Java Coding Conventions .
- It is recommended highly to follow java coding standards.
- Class names should start with uppercase letters. Classnames names should be nouns.
- If Class name is of multiple words then the first letter of the inner word must be capital letter.
- Ex : Employee, EmployeeDetails, ArrayList, TreeSet, HashSet

## 27) Explain Java Coding standards for interfaces?

- Interface should start with uppercase letters
- Interfaces names should be adjectives
- Example : Runnable, Serializable, Marker, Cloneable

## 28) Explain Java Coding standards for Methods?

- Method names should start with small letters.
- Method names are usually verbs
- If the method contains multiple words, every inner word should start with an uppercase letter.
- Ex : toString()
- Method name must be combination of verb and noun
- Ex : getCarName(),getCarNumber()

## 29) Explain Java Coding Standards for variables ?

- Variable names should start with small letters.
- Variable names should be nouns.
- Short meaningful names are recommended.
- If there are multiple words, every innerword should start with an Uppercase character.
- Ex : string,value,empName,empSalary.
- Constant names are usually nouns. Ex:MAX_VALUE, MIN_VALUE, MAX_PRIORITY, MIN_PRIORITY

## 30) What is the ‘IS-A ‘ relationship in java?

- ‘is a’ relationship is also known as inheritance.
- We can implement ‘is a’ relationship or inheritance in java using extends keyword.
- The advantage or inheritance or is a relationship is reusability of code instead of duplicating the code.
- Ex : MotorCycle is a vehicle
- Car is a vehicle Both car and motorcycle extends vehicle.

## 31) What is ‘HAS A’’ relationship in java?

- ‘Has a ‘ relationship is also known as “composition or Aggregation”.
- As in inheritance we have ‘extends’ keyword we don’t have any keyword to implement ‘Has a’ relationship in java in a specific way but we can use new keyword in another class is the concept of Has a relationship.
- The main advantage of ‘Has-A‘ relationship in java code reusability.

## 32) Difference between ‘IS-A’ and ‘HAS-A’ relationship in java?

| IS-A relationship | HAS- A RELATIONSHIP |
|---|---|
| - Is a relationship also known as inheritance | - Has a relationship also known as composition or aggregation. |
| - For IS-A relationship we use extends keyword.<br>- Ex : Car is a vehicle. | - For Has a relationship we use new keyword<br>- Ex : Car has an engine. We cannot say Car is an engine. |
| - The main advantage of inheritance is reusability of code. | - The main advantage of having a relationship is reusability of code. |

## 33) Explain about instanceof operator in java?

- Instanceof operator is used to test the object is of which type.

- Syntax : <reference expression> instanceof <destination type>
- Instanceof returns true if the reference expression is a subtype of destination type.
- Instanceof returns false if the reference expression is null.
- Since a is an Integer object it returns true.
- There will be a compile time check whether the reference expression is a subtype of destination type.
- If it is not a subtype then compile time error will be shown as Incompatible types

```
Example :
public classInstanceOfExample {
        public static void main(String[] args) {
        Integer a = newInteger(5);
        if (a instanceof java.lang.Integer) {
            System.out.println(true);
        } else {
             System.out.println(false);
        }
}
```

# 34) What does null mean in java?

- "null" is a literal value.
- When a reference variable doesn't point to any value it is assigned null.
- Example : Employee employee;
- In the above example the employee object is not instantiated so it is pointed nowhere.

# 35) Can we have multiple classes in a single file ?

- Yes we can have multiple classes in a single file but it is rarely done and not recommended.

- We can have multiple classes in File but only one class can be made public.
- If we try to make two classes in File public we get the following compilation error.
- “The public type must be defined in its own file”.

## 36) What all access modifiers are allowed for top class ? )

- For top level class only two access modifiers are allowed, public, default, abstract,final and strictfp.
- If a class is declared as public it is visible everywhere.
- If a class is declared default it is visible only in the same package.
- If a class is declared abstract we cannot create an Object of that class.
- If a class is declared as final we cannot extend that class no sub classes for that class.
- If we try to give private and protected access modifiers to class we get the below compilation error.
- “Illegal Modifier for the class only public,abstract and final are permitted”

## 37) What are packages in java?

- Package is a mechanism to group related classes ,interfaces and enums into a single module.
- Package can be declared using the following statement.
- Syntax : package Coding Convention : package name should be declared in small letters.
- package statement defines the namespace.
- The main use of package is
- 1) To resolve naming conflicts
- 2) For visibility control : We can define classes and interfaces that are not accessible outside the class.

## 38) Can we have more than one package statement in the source file ?

- We can't have more than one package statement in the source file.
- In any java program there can be at most only 1 package statement.
- We will get compilation errors if we have more than one package statement in the source file.

## 39) Can we define a package statement after an import statement in java?

- We can't define a package statement after an import statement in java.
- package statement must be the first statement in the source file.
- We can have comments before the package statement.

## 40) What are identifiers in java and rules of it?

- Identifiers are names in java programs.
- Identifiers can be class name, method name or variable name.
- Rules for defining identifiers in java:
- 1) Identifiers must start with a letter ,Underscore or dollar($) sign.
- 2) Identifiers can't start with numbers .
- 3) There is no limit on the number of characters in an identifier but it is not recommended to have more than 15 characters
- 4) Java identifiers are case sensitive.
- 5) First letter can be the alphabet ,or underscore and dollar sign.

- From the second letter we can have numbers.
- 6) We shouldn't use reserved words for identifiers in java.

# 41)What are class level modifiers :

- Whenever we are writing a java class, we have to provide some information about our class to the JVM.
- Whether this class is accessible from anywhere or not.
- Whether child class creation is possible or not.
- Whether object creation is possible or not .
- We can specify this information by declaring with appropriate modifiers.
- Only applicable modifiers for top level classes are

| |
|---|
| public |
| default |
| final |
| abstract |
| strictfp |

# 42)What are Inner classes level modifiers?

| | | |
|---|---|---|
| public | | private |
| default | | protected |
| final | + | static |
| abstract | | |
| Strictfp | | |

Access specifiers and Access modifiers.

- public, private, protected and default considered as Access specifiers.
- Except these remaining are considered as modifiers but this rule is applicable, only for old languages but not for java.

- In Java all are considered as Access modifiers only, There is no terminology like access specifiers.

# 43) What is the final access modifier in java?

- A final access modifier can be used for class, method and variables.
- The main advantage of the final access modifier is security;
- no one can modify our classes, variables and methods.
- The main disadvantage of final access modifier is we cannot implement oops concepts in java. Ex : Inheritance, polymorphism.

final class :

- A final class cannot be extended or subclassed.
- We are preventing inheritance by marking a class as final. But we can still access the methods of this class by composition.
- Ex: String class

final methods:

- Method overriding is one of the important features in java.
- But there are situations where we may not want to use this feature.
- Then we declared the method as final which will print overriding.
- To allow a method from being overridden we use a final access modifier for methods.

final variables :

- If a variable is declared as final, it behaves like a constant .
- We cannot modify the value of the final variable.
- Any attempt to modify the final variable results in compilation error.
- The error is as follows
- “The final variable cannot be assigned.”

# 44) Explain abstract classes in java?

- Whenever we don't want to create an Object of a class then we can go for an abstract keyword for a class.
- Sometimes we may come across a situation where we cannot provide implementation to all the methods in a class.
- We want to leave the implementation to a class that extends it. In such cases we declare a class as abstract.
- To make a class abstract we use keyword abstract.
- Any class that contains one or more abstract methods is declared as abstract.
- If We don't give abstract methods also we can simply class an abstract to avoid the object creation, the best example is HttpServlet which does not have any abstract methods.
- If we don't declare a class as abstract which contains abstract methods we get a compile time error. We get the following error.

"The type <class-name> must be an abstract class to define abstract methods."
Signature ; abstract class <class-name>
{
}

- For example if we take a vehicle class we cannot provide implementation to it because there may be two wheelers , four wheelers etc.
- At that moment we make vehicle class abstract. All the common features of vehicles are declared as abstract methods in vehicle class. Any class which extends vehicle will provide
- its method implementation. It's the responsibility of the subclass to provide implementation.

The important features of abstract classes are :

1) Abstract classes cannot be instantiated.
2) An abstract class contains abstract methods, concrete methods or both.

3) Any class which extends abstract class must override all methods of abstract class.
4) An abstract class can contain either 0 or more abstract methods.

Though we cannot instantiate abstract classes we can create object references . Through superclass references we can point to subclasses.

# 45) Can we create a constructor in abstract class ?

- We can create a constructor in an abstract class , it doesn't give any compilation error.
- But when we cannot instantiate class there is no use in creating a constructor for abstract class.

# 46) What are abstract methods in java?

- An abstract method is the method which does not have any body.
- Abstract method is declared with keyword abstract and semicolon in place of method body.
- Signature : public abstract void <method name>();
- Ex : public abstract void getDetails();
- It is the responsibility of the subclass to provide implementation to abstract methods defined in abstract class.

# 47) What is an exception in java?

- In java exceptions are objects.
- Any unwanted event that happens in your program is nothing but Exception.
- Exceptions are created when abnormal situations are arised in our program.
- Exceptions can be created by JVM or by our application code.
- All Exception classes are defined in java.lang.
- In other words we can say Exceptions as runtime errors.

- All exceptions are under the Throwable class which is the root of all your Exception classes.

# 48) State some situations where exceptions may arise in java?

1) Accessing an element that does not exist in an array.
(ArrayIndexOutOfBoundsException) UCE
2) Invalid conversion of number to string and string to number.
(NumberFormatException) UCE
3) Invalid casting of class
(ClassCastException) UCE
4) Any number dividing with zero is AE.
(ArithmeticException) UCE
5) null dot any operation is nothing but NPE.
(NullPointerException)UCE
6) While Loading classes regarding Oracle or MySQL into Java (class.forName()).
(ClassNotFoundException) CE
7) While Connecting to the Database through DriverManager.getConnection();
(SQLException) CE
8) While Creating a file with createNewFile() method is an IOException.
(IOException)

# 49) What is Exception handling in java?

- Exception handling is a mechanism for what to do when some abnormal situation arises in a program.
- When an exception is raised in a program it leads to termination of the program when it is not handled properly.

- The significance of exception handling comes here in order not to terminate a program abruptly and to continue with the rest of the program normally (Graceful termination of the program).
- This can be done with help of Exception handling.

# 50) What is an error in Java?

- Error is the subclass of Throwable class in java.
- When errors are caused by our program we call that as Exception, but sometimes exceptions are caused due to some environment issues such as running out of memory.
- In such cases we can't handle the exceptions. Exceptions which cannot be recovered are called as errors in java.
  Ex : Out of memory issues.

# 51) What are the advantages of Exception handling in java?

1) Separating normal code from exception handling code to avoid abnormal termination of program.
2) Categorizing into different types of Exceptions so that rather than handling all exceptions with Exception root class we can handle specific exceptions.
It is recommended to handle exceptions with specific Exceptions instead of handling with Exception root class.
3) Call stack mechanism : If a method throws an exception and it is not handled immediately, then that exception is propagated or thrown to the caller of that method. This propagation continues till it finds an appropriate exception handler ,if it finds a

handler it would be handled otherwise program terminates abruptly.

## 52) In how many ways can we do exception handling in java?

We can handle exceptions in either of the two ways :
1) By specifying a try catch block where we can catch the exception.
2) Declaring a method with a throws clause .

## 53) List out five keywords related to Exception handling ?

try → to maintain risky code
catch → to maintain handling code
finally → to maintain clean up code
throw → to handover our created exception object to JVM manually.
throws → to delegate responsibilities of exception handling to the caller.

54) Explain try and catch keywords in java?

- In the try block we define all exception causing code.
- In java try and catch forms a unit.
- A catch block catches the exception thrown by the preceding try block.
- catch block cannot catch an exception thrown by another try
- block.
- If there is no exception causing code in our program or exception is not raised in our code JVM ignores the try catch block.

Syntax :

```
try
{
}
Catch(Exception e)
{
}
```

## 54) Can we have a try block without a catch block?

- Each try block requires at least one catch block or finally block.
- A try block without catch or finally will result in compiler error.
- We can skip either catch or finally block but not both.

## 55) Can we have multiple catch blocks for a try block?

- In some cases our code may throw more than one exception.
- In such cases we can specify two or more catch clauses, each catch handling a different type of exception.
- When an exception is thrown JVM checks each catch statement in order and the first one which matches the type of exception is executed and remaining catch blocks are skipped.
- try with multiple catch blocks is highly recommended in java.
- If multiple catch blocks are present the order of catch blocks is very important and the order should be from child to parent.

# 56) Explain the importance of finally blocking in java?

Finally the block is used for cleaning up resources such as closing connections, sockets etc.
If try block executes with no exceptions then finally is called after try block without executing catch block.
If there is an exception thrown in the try block, the finally block executes immediately after the catch block.
If an exception is thrown,the finally block will be executed even if the no catch block handles the exception.
Is there any way to stop the finally block ? Yes !! using System.exit(0);

# 57) Can we have any code between try and catch blocks?

- We shouldn't declare any code between try and catch blocks.
- Catch block should immediately start after the try block.

```
try{
//code
}
System.out.println("one line of code"); // illegal
catch(Exception e){
//
}
```

# 58) Can we have any code between try and finally blocks?

- We shouldn't declare any code between try and finally block.
- finally block should immediately start after catch block.
- If there is no catch block it should immediately start after the try block.

```
try{
//code
}
System.out.println("one line of code"); // illegal
finally{
//
}
```

# 59) Can we catch more than one exception in a single catch block?

- From Java 7, we can catch more than one exception with a single catch block.
- This type of handling reduces the code duplication.
- Note : When we catch more than one exception in a single catch block , the catch parameter is implicitly final.
- We cannot assign any value to the catch parameter.

Ex : catch(ArrayIndexOutOfBoundsException || ArithmeticException e)
{

}
In the above example e is final we cannot assign any value or modify e in catch statement.

# 60) What are checked Exceptions?

1) All the subclasses of the Throwable class except Error, RuntimeException and its subclasses are checked exceptions.
2) Checked exceptions should be thrown with keyword throws or should be provided with a try catch block, else the program would not compile. We do get compilation errors.

**Examples** :

1) IOException,
2) SQlException,
3) FileNotFoundException,
4) ClassNotFoundException and
5) InterruptedException

# 61) What are unchecked exceptions in java?

- All subclasses of RuntimeException are called unchecked exceptions.
- These are unchecked exceptions because the compiler does not check if a method handles or throws exceptions.
- Program compiles even if we do not catch the exception or throws the exception.
- If an exception occurs in the program,the program terminates .
- It is difficult to handle these exceptions because there may be many places causing exceptions.

Example :
1) Arithmetic Exception
3) ArrayIndexOutOfBoundsException
4) ClassCastException
5) IndexOutOfBoundException
6) NullPointerException
7) NumberFormatException
8) StringIndexOutOfBounds

# 62) Explain differences between checked and Unchecked exceptions in java?

| Unchecked Exception | Checked Exception |
|---|---|
| 1) All the subclasses of RuntimeException are called unchecked exceptions. | All subclasses of Throwable class except RuntimeException are called as checked exceptions |
| 2) Unchecked exceptions need not be handled at compile time | Checked Exceptions need to be handled at compile time. |
| 3) These exceptions arise mostly due to coding mistakes in our program | These exceptions arise mostly due to specific method calls and Object creations in our program |
| 4) ArrayIndexOutOfBoundsException, ClassCastException, IndexOutOfBoundException | SqlException, FileNotFoundException,ClassNotFoundException |

# 63) What is default Exception handling in java?

- When JVM detects exception-causing code, it constructs a new exception handling object by including the following information.
    1) Name of Exception
    2) Description about the Exception
    3) Location of Exception.
- After creation of an object by JVM it checks whether there is exception handling code

or not.

- If there is exception handling code then exception handles and continues the program.
- If there is no exception handling code JVM gives the responsibility of exception handling to the default handler and terminates abruptly.
- Default Exception handler displays description of exception,prints the stack trace and location of exception and terminates the program.
- Note : The main disadvantage of this default exception handling is that the program terminates abruptly.

# 64) Explain the throw keyword in java?

- Generally JVM throws the exception and we handle the exceptions by using try catch blocks.
- But there are situations where we have to throw user defined exceptions or runtime exceptions.
- In such cases we use throw keyword to throw exceptions explicitly.
- Syntax : throw throwableInstance;
- Throwable instances must be of type throwable or any of its subclasses.
- After the throw statement execution stops and subsequent statements are not executed. Once an exception object is thrown JVM checks if there is any catch block to handle the exception.
- If not then the next catch statement till it finds the appropriate handler.
- If the appropriate handler is not found ,then the default exception handler halts the program and prints the description and location of exception.
- In general we use throw keyword for throwing user defined or customized exceptions.

# 65) Can we write any code after the statement?

- After the throw statement JVM stops execution and subsequent statements are not executed.
- If we try to write any statement after a throw we do get a compile time error saying

unreachable code.

# 66) Explain the importance of the “throws” keyword in java?

throws statement is used at the end of method signature to indicate that an exception of a given type may be thrown from the method.
The main purpose of throws is to delegate responsibility of exception handling to the caller methods, in the case of checked exceptions.
In the case of unchecked exceptions, it is not required to use throws.
We can use throws keyword only for throwable types otherwise compile time error saying incompatible types.
An error is unchecked , it is not required to handle by try catch or by throws.
Syntax : Class Test{ Public static void main(String args[]) throws IE { } }
Note : The method should throw only checked exceptions and subclasses of checked exceptions.
It is not recommended to specify exception superclasses in the throws class when the actual exceptions thrown in the method are instances of their subclass.

# 67) Explain the importance of finally over-returning statements?

- finally block is more important than a return statement when both are present in a program.
- For example if there is any return statement present inside try or catch block, and finally block is also present, the first finally statement will be executed and then return statement will be considered.

Explain a situation where the finally block will not be executed?

- finally, the block will not be executed whenever JVM shutdowns.
- If we use system.exit(0) in try statement finally block if present will not be executed.

# 68)Can we use catch statements for checked exceptions?

If there is no chance of raising an exception in our code then we can't declare a catch block for handling checked exceptions .
This raises compile time error if we try to handle checked exceptions when there is no possibility of causing exception.

# 69) What are user defined exceptions?

- To create customized error messages we use user defined exceptions. We can create user defined
- exceptions as checked or unchecked exceptions.
- We can create user defined exceptions that extend Exception class or subclasses of checked exceptions so that user defined exceptions become checked.
- User Defined exceptions can extend RuntimeException to create user defined unchecked exceptions.
- Note : It is recommended to keep our customized exception class as unchecked,i.e we need to extend
- Runtime Exception class but not Exception class.

## 67) Can we rethrow the same exception from the catch handler?

Yes we can rethrow the same exception from our catch handler.
If we want to rethrow checked exceptions from a catch block we need to declare that exception.

## 68) Can we have nested try statements in java?

- Yes, try statements can be nested.
- We can declare try statements inside the block of another try statement or catch or finally.

## 69) Explain the importance of a throwable class and its methods?

- Throwable class is the root class for Exceptions.
- All exceptions are derived from this throwable class.
- The two main subclasses of Throwable are Exception and Error.

The three methods defined in throwable class
are :

- 1) void printStackTrace() :
  This prints the exception information in the following format :
  Name of the exception, description followed by stack trace.
- 2) getMessage()
  This method prints only the description of Exception.
- 3) toString():
  It prints the name and description of Exception.

# 70) Explain when ClassNotFoundException will be raised ?

When JVM tries to load a class by its string name, and couldn't able to find the class classNotFoundException will be thrown.

An example for this exception is when class name is misspelled
and when we try to load the class by string name hence class cannot be found which raises ClassNotFoundException.

one more example : When we call class.forName("OracleDriver");

# 71) Explain when NoClassDefFoundError will be raised ?

This error is thrown when JVM tries to load the class but no definition for that class is found NoClassDefFoundError will occur.

The class may exist at compile time but unable to find at runtime.

This might be due to misspelled class name at command line, or classpath is not specified properly , or the
class file with byte code is no longer available.

# Java Interview questions on threads

## 72) What is the process ?

A process is a program in execution.
Every process has its own memory space.
Processes are heavy weight and require their own address space. One or more threads make a process.

## 73) What is thread in java?

Thread is a separate path of execution in a program.
Threads are
1) Light weight
2) They share the same address space.
3) creating thread is simple when compared to process because creating thread requires less resources when compared to process
4) Threads exist in process. A process have at least one thread

## 74) Difference between process and thread ?

| Process | Thread |
|---|---|
| Program in execution. | Separate path of execution in program. One or more threads is called a process. |
| Processes are heavy weight. | Threads are lightweight. |
| Processes require separate address space. | Threads share the same address space. |
| Interprocess communication is expensive. | Inter Thread communication is less expensive compared to processes. |

| | |
|---|---|
| Context switching from one process to another is costly. | Context switching between threads is low cost. |

# 75) What is multitasking ?

Multitasking means **performing more than one activity at a time** on the computer.

Example Using spreadsheet and using calculator at same time.

# 76) What are different types of multitasking?

There are two different types of multitasking :

1) Process based multitasking
2) Thread based multitasking

**Process based multitasking** : It allows to run two or more programs concurrently.
In process based multitasking a process is the smallest part of code .
Example : Running Ms word and Ms powerpoint at a time.

**Thread based multitasking** : It allows parts of a program to run concurrently.
Example : Formatting the text and printing word document at same time .
Java supports thread based multitasking and provides built in support for multithreading.

# 77) What are the benefits of multithreaded programming?

Multithreading enables the use of idle time of cpu to another thread which results in faster execution of the program.

In a single threaded environment each task has to be completed before proceeding to the next task making the cpu idle.

# 78) Explain thread in java?

1) Thread is an independent path of execution within a program.
2) A thread consists of three parts: Virtual Cpu, Code and data.
3) At run time threads share code and data i.e they use the same address space.
4) Every thread in java is an object of java.lang.Thread class

# 79) List Java API that supports threads?

**java.lang.Thread :** This is one of the ways to create a thread.
By extending Thread class and overriding run() we can create thread in java.

**java.lang.Runnable :** Runnable is an interface in java. By implementing a runnable interface and overriding run() we can create thread in java.

**java.lang.Object :** Object class is the super class for all the classes in java. In object class we have three methods wait(), notify(), notifyAll() that support threads.

**java.util.concurrent :** This package has classes and interfaces that supports concurrent programming.
Ex : **Executor** interface, Future task class etc.

## 80) Explain about the main thread in java?

Main thread is important because :
1) All the child threads spawn from the main thread.
2) Main method is the last thread to finish execution.
When JVM calls main method() it starts a new thread. Main() method is temporarily stopped while the a new thread starts running.

## 81) In how many ways can we create threads in java?

We can create threads in java by any of the two ways :
1) By extending Thread class
2) By Implementing Runnable interface.

## 82) Explain creating threads by implementing Runnable class?

This is the first and foremost way to create threads .
By implementing a runnable interface and implementing run() method we can create new threads.

Method signature : public void run()
Run is the starting point for execution for another thread within our program.

Example :

```
public class MyClass implements Runnable {
@Override
public void run() {
// T
}
}
```

## 83) Explain creating threads by extending Thread class ?

We can create a thread by extending Thread class. The class which extends Thread class must override the run() method.

Example :

```
public class MyClass extends Thread {
@Override
public void run() {

// Starting point of Execution
}
}
```

## 84) Which is the best approach for creating thread ?

The best way for creating threads is to implement a runnable interface.
When we extend Thread class we can't extend any other class.
When we create a thread by implementing a Runnable interface we can implement Runnable interface.
In both ways we have to implement the run() method.

# 85) Explain the importance of thread scheduler in java?

Thread scheduler is part of the JVM used to determine which thread to run at this moment when there are multiple threads. Only threads in runnable state are chosen by scheduler.

Thread scheduler first allocates the processor time to the higher priority threads.

To allocate microprocessor time in between the threads of the same priority, thread scheduler follows round robin fashion

# 86) Explain the life cycle of thread?

A thread can be in any of the five states :

1) **New** : When the instance of thread is created it will be in New state.
**Ex : Thread t= new Thread();**
In the above example it is in a new state. The thread is created but not in active state.
To make it active we need to call the start() method on it.

2) **Runnable state** : A thread can be in the runnable state in either of the following two ways
a) When the start method is invoked or
b) A thread can also be in a runnable state after coming back from a blocked or sleeping or waiting state.

3) **Running state** : If the thread scheduler allocates cpu time, then the thread will be in running state.

4) **Waited /Blocking/Sleeping state**:
In this state the thread can be made temporarily inactive for a short period of time. A thread can be in the above state in any of the following ways:
1) The thread waits to acquire the lock of an object.
2) The thread waits for another thread to complete.

3) The thread waits for notification of another thread.
5) Dead State : A thread is in dead state when thread's run method execution is complete.

It dies automatically when thread's run method execution is completed and the thread object will be garbage collected.

## 87) Can we restart a dead thread in java?

If we try to restart a dead thread by using the start method we will get **run time exception** since the thread is not alive.

## 88) Can one thread block the other thread?

No one thread cannot block the other thread in java.
It can block the current thread that is running.

## 89) Can we restart a thread already started in java?

A thread can be started in java using start() method in java.
If we call start method a second time once it is started it will cause a RunTimeException(**IllegalThreadStateException**).
A runnable thread cannot be restarted.

## 90) What happens if we don't override the run method ?

If we don't override the run method .
Then the default **implementation of Thread class run() method will be executed** and hence the thread will never be in runnable state.

## 91) Can we overload the run() method in java?

We can overload the run method but Thread class start method will always call run method with no arguments.

But the overloaded method will not be called by the start method.
We have to explicitly call this start() method.

## 92) What is a lock or purpose of locks in java?

**Lock**, also called **monitor**, is used to prevent access to a shared resource by multiple threads.

A lock is associated with a shared resource. Whenever a thread wants to access a shared resource it must first acquire a lock .
If a lock has been acquired by another, it can't access that shared resource.

At this moment the thread has to wait until another thread releases the lock on a shared resource. To lock an object we use **synchronization** in java.
A lock protects a section of code allowing only one thread to execute at at a time.

## 93) In how many ways can we do synchronization in java?

There are two ways to do synchronization in java:
1) Synchronized methods
2) Synchronized blocks
To do synchronization we use synchronized keyword.

# 94) What are synchronized methods ?

If we want a method of an object to be accessed by a single thread at a time we declare that method with a synchronized keyword.

**Signature** :
**public synchronized void methodName(){}**

To execute synchronized method first lock has to be acquired on that object.

Once a synchronized method is called, lock will be automatically acquired on that method when no other thread has lock on that method.

Once a lock has been acquired then the synchronized method gets executed.

Once synchronized method execution completes automatically lock will be released.

The prerequisite to execute a synchronized method is to acquire lock before method execution.

If there is a lock already acquired by any other thread it waits till the other thread completes

# 95) When do we use synchronized methods in java?

If multiple threads tries to access a method where method can manipulate the state of object , in such scenario we can declare a method as synchronized.

## 96) When a thread is executing synchronized methods , then is it possible to execute other synchronized methods simultaneously by other threads?

No it is not possible to execute synchronized methods by other threads when a thread is inside a synchronized method.

## 97) When a thread is executing a synchronized method , then is it possible for the same thread to access other synchronized methods of an object ?

Yes it is possible for thread executing a synchronized method to execute another synchronized method of an object.

```
public synchronized void methodName()
{
}
```

To execute synchronized method first lock has to be acquired on that object.

Once a synchronized method is called, lock will be automatically acquired on that method when no other thread has lock on that method.

Once lock has been acquired then the synchronized method gets executed.

Once synchronized method execution completes automatically lock will be released.

The prerequisite to execute a synchronized method is to acquire lock before method execution.

If there is a lock already acquired by any other thread it waits till the other thread completes

# 98) What are synchronized blocks in java?

Synchronizing a few lines of code rather than a complete method with the help of synchronized **keyword** are called synchronized blocks.

**Signature :Synchronized (object reference){// code}**

# 99) When do we use synchronized blocks and advantages of using synchronized blocks?

If very few lines of code require synchronization then it is recommended to use synchronized blocks.
The main advantage of synchronized blocks over synchronized methods is it reduces the waiting time of threads and improves performance of the system.

# 100) What is class level lock ?

Acquiring lock on the class instance rather than the object of the class is called class level lock.

The difference between class level lock and object level lock is in class level lock lock is acquired on class .class instance and in object level lock ,lock is acquired on object of class.

# 101) Can we synchronize static methods in java?

Every class in java has a unique lock associated with it.

If a thread wants to execute a static synchronized method it needs to acquire first class level lock.

When a thread is executing static synchronized method no other thread can execute static synchronized method of class since lock is acquired on class.

But it can execute the following methods simultaneously :
1) Normal static methods
2) Normal instance methods
3) synchronize instance methods
Signature : synchronized(Classname.class){}

# 102) Can we use synchronized blocks for primitives?

Synchronized blocks are applicable only for objects if we try to use synchronized blocks for primitives we get compile time errors.

# 103) What are thread priorities and importance of thread priorities in java?

When there are several threads in waiting, thread priorities determine which thread to run. In the Java programming language every thread has a priority.

A thread inherits the priority of its parent thread.

By default thread has a normal priority of 5.

Thread scheduler uses thread priorities to decide when each thread is allowed to run.
Thread scheduler runs higher priority threads first.

# 104) Explain different types of thread priorities ?

Every thread in java has priorities in between 1 to 10. By default priority is 5 (Thread.NORM_PRIORITY). The maximum priority would be 10 and minimum would be 1.Thread class
defines the following constants(static final variables) to define properties.
Thread. MIN_PRIORITY = 1;
Thread.NORM_PRIORITY=5;
Thread. MAX_PRIORITY=10;

# 105) How to change the priority of thread or how to set priority of thread?

Thread class has a set method to set the priority of thread and get method to get the priority of the thread.

Signature : final void setPriority(int value);

The setPriority() method is a request to JVM to set the priority.
JVM may or may not oblige the request.

We can get the priority of the current thread by using the getPriority() method of Thread class.

```
final int getPriority()
{
}
```

## 106) If two threads have the same priority, which thread will be executed first ?

We are not guaranteed which thread will be executed first when there are threads with equal priorities in the pool. It depends on the thread scheduler to which thread to execute.

The scheduler can do any of the
following things :
1) It can pick any thread from the pool and run it till it completes.
2) It can give equal opportunity for all the threads by time slicing.

## 107) What all methods are used to prevent thread execution ?

There are three methods in the Thread class which prevent execution of thread.
1) yield()
2) join()
3) sleep()

# 108) Explain yield() method in thread class ?

Yield() method makes the current running thread move into a runnable state from the running state giving a chance to remaining threads of equal priority which are in waiting state.

yield() makes current thread to sleep for a specified amount of time.

There is no guarantee that moving a current running thread from runnable to running state. It all depends on the thread scheduler; it doesn't guarantee anything.
synchronized(Classname.class){ }

# 109) Is it possible for a yielded thread to get a chance for its execution again ?

**Yield**() causes the current thread to sleep for a specified amount of time giving opportunity for other threads of equal priority to execute.

Thread scheduler decides whether it gets a chance for execution again or not.
It all depends on the mercy of thread scheduler.

# 110) Explain the importance of join() method in thread class?

A thread can invoke the join() method on another thread to wait for another thread to complete its execution.

Assume we have two threads, t1 and t2 threads . A running thread t1 invokes join() on thread t2 and t1 thread will wait in a waiting state until t2 completes.

Once t2 completes the execution, t1 will continue.

join() method throws Interrupted Exception so whenever we use join() method we should handle Interrupted Exception by throws or by using try catch block.

Signature
public final void join() throws InterruptedException
{
}

public final synchronized void join(long millis) throws InterruptedException { }

public final synchronized void join(long millis, int nanos) throws InterruptedException {
}

# 111) Explain purpose of sleep() method in java?

sleep() method causes the current running thread to sleep for a specified amount of time.

sleep() method is the minimum amount of the time the current thread sleeps but not the exact amount of time.

Signature :
public static native void sleep(long millis) throws InterruptedException { }

public static void sleep(long millis, int nanos) throws InterruptedException { }

## 112) Assume a thread has lock on it, calling sleep() method on that thread will release the lock?

Calling sleep() method on thread which has lock does not affect.
Lock will not be released though the thread sleeps for a specified amount of time.

## 113) Can sleep() method cause another thread to sleep?

No sleep() method causes current thread to sleep not any other thread.

## 114) Explain about the interrupt() method of thread class ?

Thread class interrupt() method is used to interrupt current thread or another thread.

It does not mean the current thread will stop immediately, it is a polite way of telling or requesting to continue your present work.

That is the reason we may not see the impact of interrupt calls immediately.

Initially the thread has a boolean property(interrupted status) false.

So when we call interrupt() method status would be set to true.

This causes the current thread to continue its work and does not have impact immediately.

If a thread is in sleeping or waiting status (i.e thread has executed wait () or sleep() method) thread gets interrupted it stops what it is doing and throws an interrupted

exception.

This is the reason we need to handle interrupted exceptions with throws or try/ catch blocks.

# 115) Explain about inter thread communication and how it takes place in java?

Usually threads are created to perform different unrelated tasks but there may be situations where they may perform related tasks.
Inter Thread communication in java is done with the help of following three methods :

1) wait()
2) notify()
3) notifyAll()

# 116) Explain wait(), notify() and notifyAll() methods of object class ?

**wait**() : wait() method() makes the current thread sleep and releases the lock until some other thread acquires the lock and calls notify().

**notify**() :notify() method wakes up the thread that called wait on the same object.

**notfiyAll**() :notifyAll() method wakes up all the threads that are called wait() on the same object. The highest priority threads will run first.

All the above three methods are in object class and are called only in synchronized context.

All the above three methods must handle InterruptedException by using throws clause or by using try catch clause.

## 117) Explain why wait() , notify() and notifyAll() methods are in Object class rather than in thread class?

First to know why they are in object class we should know what wait(), notify(), notifyAll() methods do.

wait() , notify(), notifyAll() methods are object level methods that are called on the same object.

wait(), notify(), notifyAll() are called on an shared object so to they are kept in object class rather than thread class.

## 118) Explain IllegalMonitorStateException and when it will be thrown?

IllegalMonitorStateException is thrown when wait(), notify() and notifyAll() are called in non synchronized context.

Wait(), notify(),notifyAll() must always be called in synchronized context otherwise we get this run time exception.

# 119) when wait(), notify(), notifyAll() methods are called does it releases the lock or holds the acquired lock?

wait(), notify(), notifyAll() methods are always called in synchronized context.
When these methods are called in a synchronized context.

So when they enter first in synchronized context the thread acquires the lock on the current object.

When wait(), notify(), notifyAll() methods are called and lock is released on that object.

# 120) Explain which of the following methods releases the lock when yield(), join(),sleep(),wait(),notify(), notifyAll() methods are executed?

| Method | Releases lock (Yes or No) |
|---|---|
| **yield()** | **NO** |
| **sleep()** | **NO** |
| **join()** | **NO** |
| **wait()** | **Yes** |
| **notify()** | **Yes** |
| **notifyAll()** | **Yes** |

# 121) What are thread groups?

Thread Groups are group of threads and other thread groups. It is a way of grouping threads so that actions can be performed on a set of threads for easy maintenance and security purposes.

For example we can start and stop all thread groups.

We rarely use thread group class. By default all the threads that are created belong to default thread group of the main thread. Every thread belongs to a thread group. Threads that belong to a particular thread group cannot modify threads belonging to another thread group.

# 122) What are thread local variables ?

Thread local variables are variables associated with a particular thread rather than object.

We declare a ThreadLocal object as a private static variable in a class.

Everytime a new thread accesses an object by using getter or setter we are accessing a copy of the object.

Whenever a thread calls get or set method of ThreadLocal instance a new copy is associated with a particular object.

# 124) What are daemon threads in java?

Daemon threads are threads which run in the background.

These are service threads and work for the benefit of other threads.

**Garbage collector is one of the good examples** for daemon threads.

By default all threads are non daemon. Daemon nature of a thread can be inherited.

**If parent thread is daemon , child thread also inherits daemon nature of thread.**

# 125) How to make a non daemon thread as daemon?

By default all threads are non daemon. We can make the non daemon nature of thread to daemon by using **setDaemon() method.**

The important point to note here is that we can call setDaemon() only before the start() method is called on it.

If we call setDaemon() after start() method an **IllegalThreadStateException** will be thrown.

# 126) Can we make main() thread as daemon?

Main thread is always non daemon. We cannot change the non daemon nature of the main thread to daemon.

# Interview questions on Nested classes and inner classes

## 127) What are nested classes in java?

Class declared within another class is defined as a nested class.

There are two types of nested classes in java.
1) Static nested class
2) Non static nested class
A static nested class has a static keyword declared before class definition.

## 128) What are inner classes or non static nested classes in java?

Nested classes without any static keyword declaration in class definition are defined as non static nested
classes. Generally non-static nested classes are referred to as inner classes.
There are three types of inner classes in java :
1) Local inner class
2) Member inner class
3) Anonymous inner class

## 129) Why to use nested classes in java?

(or)

# 130) What is the purpose of nested class in java?

1) Grouping of related classes
Classes which are not reusable can be defined as inner class instead of creating inner class.
For example : We have a submit button upon click of submit button we need to execute some code. This code is related only to that class and cannot be reused for other classes . Instead of creating a new class we
can create inner class

2) To increase encapsulation :
Inner class can access private members of outer class.so by creating getter and setter methods for private variables , the outside world can access these variables.

But by creating inner class private variables can be accessed only by inner class.

3) Code readable and maintainable :
Rather than creating a new class we can create an inner class so that it is easy to maintain.

4) Hiding implementation :
Inner class helps us to hide implementation of class.

# 131) Explain about static nested classes in java?

When a static class is defined inside an enclosing class we define that as a nested class. Static nested classes are not inner classes. Static nested classes can be instantiated without an instance of outer class.

A static nested class does not have access to instance variables and non static methods of outer class.

## 132) How to instantiate static nested classes in java?

We can access static members and static methods of outer class without creating any instance of outer class. Syntax for instantiating Static nested class :
OuterclassName.StaticNestedClassName ref=new
OuterclassName.StaticNestedClassName();

## 133) Explain about method local inner classes or local inner classes in java?

Nested classes defined inside a method are local inner classes.

We can create objects of the local inner class only inside the method where class is defined.

A local inner classes exist only when method is invoked and goes out of scope when the method returns.

## 134) Explain features of the local inner class?

1) Local inner class does not have any access modifier.
2) We cannot use static access modifiers for local inner classes. But we can use abstract and final for local inner classes.
3) We cannot declare static members inside local inner classes.
4) We can create objects of the local inner class only inside the method where the class is defined.

5) Method local inner classes can only access final variables declared inside a method.
6) Method local inner classes can be defined inside loops(for,while) and blocks such as if etc