# 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 gets executed exactly once when the class is loaded

. Static blocks are executed even before the constructors are executed.

# How to call one constructor from the other constructor ?

Within the same class if we want to call one constructor from other we use this() method. Based on the number of parameters we pass appropriate 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

# What is method overriding in java ?

If we have methods with same signature (same name, same signature, same return type) in super class and subclass then we say

subclass method is overridden by superclass. When to use overriding in java

If we want same method with different behavior in superclass and subclass then we go for overriding. When we call overridden method with subclass reference subclass method is called hiding the superclass method.

# What is super keyword in java ?

Variables and methods of super class can be overridden in subclass . In case of overriding , a subclass object call its own variables and methods. Subclass cannot access the variables and methods of superclass because the overridden variables or methods hides the methods and variables of super class. 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 constructor.
2. Second one is to call super class variables, methods. Super if present must be the first statement.

# Difference between method overloading and method overriding in java ?

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| 1) Method Overloading occurs with in the same class | Method Overriding occurs between  two classes superclass and subclass |
| 2) Since it involves with only one class inheritance is not involved. | Since method overriding occurs between superclass and subclass inheritance is involved. |
| 3)In overloading return type need not be the same | 3) In overriding return type must be same. |
| 4) Parameters must be different when we do overloading | 4) Parameters must be same. |
| 5) Static polymorphism can be achieved using method overloading | 5) Dynamic polymorphism can be achieved using method overriding. |
| 6) In overloading one method can’t hide the another | 6) In overriding subclass method hides that of the superclass method. |

# Difference between abstract class and interface ?

|  |  |
| --- | --- |
| Interface | Abstract Class |
| 1) Interface contains only abstract methods | 1) Abstract class can contain abstract methods, concrete methods or both |
| 2) Access Specifiers for methods in interface must be public | 2) Except private we can have any access specifier for methods in abstract class. |
| 3) Variables defined must be public , static , final | 3) Except private variables can have any access specifiers |
| 4) Multiple Inheritance in java is implemented using interface | 4)We cannot achieve multiple inheritance using abstract class. |
| 5) To implement an interface we use implements keyword | 5)To implement an interface we use implements keyword |

# Why java is platform independent?

The most unique feature of java is platform independent. In any programming language source code is compiled in to 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 windows environment can also be executed in Linux environment. This makes java platform independent.

# What is method overloading in java ?

A class having two or more methods with 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 compiler executes this method.

Compiler decides which method to call at compile time. By using 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.

# What is difference between c++ and Java ?

|  |  |
| --- | --- |
| Java | C++ |
| 1) Java is platform independent | C++ is platform dependent. |
| 2) There are no pointers in java | There are pointers in C++. |
| 3) There is no operator overloading in java | C ++ has operator overloading. |
| 4) There is garbage collection in java | There is no garbage collection |
| 5) Supports multithreading | Doesn’t support multithreading |
| 6) There are no templates in java | There are templates in java |
| 7) There are no global variables in java | There are global variables in c++ |

# What is JIT compiler ?

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

# What is bytecode in java ?

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

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

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

# What is a class ?

Classes are fundamental or basic unit in Object Oriented Programming .A class is kind of blueprint or template for objects. Class defines variables, methods. A class tells what type of objects we are creating. For example take 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 class keyword. A class definition must be saved in class file that has same as class name. File name must end with .java extension.

# What is an object ?

An Object is instance of class. A class defines 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.

# What is 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.

# What is encapsulation ?

The process of wrapping or putting up of data in to a single unit class and keeps 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 car we have many parts which is not required for driver to know what all it consists inside. 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.

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

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

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

# 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 constructor in java. We have two types of constructors

Default Constructor Parameterized Constructor Signature : public classname()

{

}

Signature : public classname(parameters list)

{

}

# Difference between overriding and overloading in java?

|  |  |
| --- | --- |
| **Overriding** | **Overloading** |
| In overriding method names must be same | In overloading method names must be same |
| Argument List must be same | Argument list must be different at least order of arguments. |
| Return type can be same or we can return covariant type. From 1.5 covariant types are allowed | Return type can be different in overloading. |
| We cant increase the level of checked exceptions. No restrictions for unchecked exceptions | In overloading different exceptions can be thrown. |
| A method can only be overridden in subclass | A method can be overloaded in same class or subclass |
| Private, static and final variables cannot be overridden. | Private , static and final variables can be overloaded. |
| In overriding which method is called is decided at runtime based on the type of object referenced at run time | In overloading which method to call is decided at compile time based on reference type. |
| Overriding is also known as Runtime polymorphism, dynamic polymorphism or late binding | Overloading is also known as Compile time polymorphism, static polymorphism or early binding. |

# What is ‘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 : Motor cycle is a vehicle

Car is a vehicle Both car and motorcycle extends vehicle.

# 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. The main advantage of ‘Has-A‘ relationship in java code reusability.

# 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 uses extends keyword | For Has a relationship we use new keyword |
| Ex : Car is a vehicle. | 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 has a relationship is reusability of code. |

# What are identifiers in java?

Identifiers are names in java program. Identifiers can be class name, method name or variable name. Rules for defining identifiers in java:

1. Identifiers must start with letter, Underscore or dollar($) sign.
2. Identifiers can’t start with numbers .
3. There is no limit on number of characters in identifier but not recommended to have more than 15 characters
4. Java identifiers are case sensitive.
5. First letter can be alphabet ,or underscore and dollar sign. From second letter we can have numbers

.

1. We shouldn’t use reserve words for identifiers in java.

# What is an exception in java?

In java exception is an object. Exceptions are created when an abnormal situations are raised 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 Exception as run time error.

# State some situations where exceptions may arise in java?

1. Accessing an element that does not exist in array.
2. Invalid conversion of number to string and string to number. (NumberFormatException)
3. Invalid casting of class (Class cast Exception)
4. Trying to create object for interface or abstract class (Instantiation Exception)

# What is Exception handling in java?

Exception handling is a mechanism what to do when some abnormal situation arises in program. When an exception is raised in program it leads to termination of 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 program normally. This can be done with help of Exception handling.

# 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 some times 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.

# What are advantages of Exception handling in java?

1. Separating normal code from exception handling code to avoid abnormal termination of program.
2. Categorizing in to different types of Exceptions so that rather than handling all exceptions with Exception root class we can handle with specific exceptions. It is recommended to handle exceptions with specific Exception 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 handler it would be handled otherwise program terminates abruptly.

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

We can handle exceptions in either of the two ways :

1. By specifying try catch block where we can catch the exception.
2. Declaring a method with throws clause .

# List out five keywords related to Exception handling ?

1. Try
2. Catch
3. throw
4. throws
5. finally

# Explain try and catch keywords in java?

In try block we define all exception causing code. In java try and catch forms a unit. A catch block catches the exception thrown by 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)

{

}

# Can we have try block without 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 of catch or finally block but not both.

# Can we have multiple catch block for a try block?

In some cases our code may throw more than one exception. In such case we can specify two or more catch clauses, each catch handling 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 execution and remaining catch blocks are skipped.

Try with multiple catch blocks is highly recommended in java.

If try with multiple catch blocks are present the order of catch blocks is very important and the order should be from child to parent.

# Explain importance of finally block in java?

Finally block is used for cleaning up of 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 exception thrown in try block finally block executes immediately after catch block.

If an exception is thrown, finally block will be executed even if the no catch block handles the exception.

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

We shouldn’t declare any code between try and catch block. Catch block should immediately start after try block.

try{

//code

}

System.out.println(“one line of code”); // illegal catch(Exception e){

//

}

# 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 try block.

try{

//code

}

System.out.println(“one line of code”); // illegal finally{

//

}

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

From Java 7, we can catch more than one exception with single catch block. This type of handling reduces the code duplication.

Note : When we catch more than one exception in single catch block , catch parameter is implicitly final. We cannot assign any value to 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.

# What are checked Exceptions?

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

Examples :

1. IOException,
2. SQlException,
3. FileNotFoundException,
4. InvocationTargetException,
5. CloneNotSupportedException
6. ClassNotFoundException
7. InstantiationException

# What are unchecked exceptions in java?

All subclasses of RuntimeException are called unchecked exceptions. These are unchecked exceptions because compiler does not checks 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, program terminates . It is difficult to handle these exceptions because there may be many places causing exceptions.

Example : 1) Arithmetic Exception

1. ArrayIndexOutOfBoundsException
2. ClassCastException
3. IndexOutOfBoundException
4. NullPointerException
5. NumberFormatException
6. StringIndexOutOfBounds
7. UnsupportedOperationException

# Explain differences between checked and Unchecked exceptions in java?

|  |  |
| --- | --- |
| Unchecked Exception | Checked Exception |
| 1) All the subclasses of RuntimeException are called unchecked exception. | 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. |  |
| 4) ArrayIndexOutOfBoundsException, ClassCastException, IndexOutOfBoundException | SqlException, FileNotFoundException,ClassNotFoundException |

# 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 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 give the responsibility of exception handling to 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 program terminates abruptly.

# Explain throw keyword in java?

Generally JVM throws the exception and we handle the exceptions by using try catch block. But there are situations where we have to throw userdefined exceptions or runtime exceptions. In such case we use throw keyword to throw exception explicitly.

Syntax : throw throwableInstance;

Throwable instance must be of type throwable or any of its subclasses.

After the throw statement execution stops and subsequent statements are not executed. Once exception object is thrown JVM checks is there any catch block to handle the exception. If not then the next catch statement till it finds the appropriate handler. If appropriate handler is not found ,then 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 exception.

# Can we write any code after throw statement?

After throw statement jvm stop execution and subsequent statements are not executed. If we try to write any statement after throw we do get compile time error saying unreachable code.

# Explain importance of 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 keyword is to delegate responsibility of exception handling to the caller methods, in the case of checked exception.

In the case of unchecked exceptions, it is not required to use throws keyword.

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 super classes in the throws class when the actual exceptions thrown in the method are instances of their subclass.

# Explain the importance of finally over return statement?

finally block is more important than 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 first finally statement will be executed and then return statement will be considered.

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

Finally 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.

# Can we use catch statement for checked exceptions?

If there is no chance of raising an exception in our code then we can’t declare 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.

# 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 userdefined exception becomes checked.

Userdefined exceptions can extend RuntimeException to create userdefined 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.

# Can we rethrow the same exception from catch handler?

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

# Can we have a method name same as class name in java?

Yes we can have method name same as class name it won’t throw any compilation error but it shows a warning message that method name is same as class name.

# Can we override constructors in java?

Only methods can be overridden in java. Constructors can’t be inherited in java. So there is no point of overriding constructors in java.

# Can Static methods access instance variables in java?

No.Instance variables can’t be accessed in static methods. When we try to access instance variable in static method we get compilation error. The error is as follows:

Cannot make a static reference to the non static field name

# How do we access static members in java?

Instance variables and instance methods can be accessed using reference variable . But to access static variables or static methods we use Class name in java.

# Explain wrapper classes in java?

Converting primitives to objects can be done with the help of wrapper classes. Prior to java 1.5 we use Wrapper classes to convert primitives to objects. From java 1.5 we have a new feature autoboxing which is used to convert automatically primitives to objects but in wrapper classes programmer has to take care of converting primitives to objects.

Wrapper classes are immutable in java. Once a value is assigned to it we cannot change the value.

# Explain different types of wrapper classes in java?

For every primitive in java we have corresponding wrapper class. Here are list of wrapper classes available in java.

|  |  |
| --- | --- |
| Primitive | Wrapper Class |
| Boolean | Boolean |
| int | Integer |
| float | Float |
| char | Character |
| byte | Byte |
| long | Long |
| short | Short |

# Define interface in java?

Interface is collection of abstract methods and constants. An interface is also defined as pure or 100 percent abstract class. Interfaces are implicitly abstract whether we define abstract access modifier or not. A class implementing interface overrides all the abstract methods defined in interface. Implements keyword is used to implement interface.

# What is the purpose of interface?

Interface is a contract . Interface acts like a communication between two objects. When we are defining interface we are defining a contract what our class should do but not how it does. An interface doesn’t define what a method does. The power of interface lies when different classes that are unrelated can implement interface. Interfaces are designed to support dynamic method resolution at run time.

# Explain features of interfaces in java?

1. All the methods defined in interfaces are implicitly abstract even though abstract modifier is not declared.
2. All the methods in interface are public whether they are declared as public or not.
3. variables declared inside interface are by default public, static and final.
4. Interfaces cannot be instantiated.
5. we cannot declare static methods inside interface.
6. ‘ implements’ keyword is used to implement interface.
7. Unlike class, interface can extend any number of interfaces.
8. We can define a class inside interface and the class acts like inner class to interface.
9. An interface can extend a class and implement an interface
10. Multiple inheritance in java is achieved through interfaces.

# What is collections framework ?

A framework is set of classes and interfaces to build a functionality. Java collections framework provides set of interfaces and classes for storing and manipulating collections. Collection framework contains classes and interfaces in java.util package and java.util.concurrent packages.

Advantages or benefits of Collections framework :

1. High performance
2. Using this framework we can create different types of collections
3. We can create our own collection and we can extend a collection.
4. Reduces programming effort.
5. Increases speed and quality : Collections framework provides high performance, implementations of useful data structures and algorithms.

# What is collection ?

A collection is a container which holds group of objects. Collection provides a way to manage objects easily. Collections manages group of objects as single unit.

Examples include list of strings, integers etc.

Here are few basic operations we do on collections :

1. Adding objects to collection.
2. Removing or deleting objects from collection.
3. Retrieving object from collection.
4. Iterating collection.

# Difference between collection, Collection and Collections in java?

collection : represent group of objects where objects are stored.

Collection : This is one of the core interface which provides basic functionality for collection. Collections : Collections contains some utility static methods that operate on collections.

# Explain about Collection interface in java ?

Collection is the fundamental and root interface in Collections framework. Collection extends Iterable interface and inherits iterator method which returns Iterator object.

# What is vector?

Vector is similar to arraylist used for random access. Vector is a dynamic array like arraylist.

vector size increases or decreases when elements are added and removed . Vector is synchronized .

vector and Hashtable are the only collections since 1.0. Rest of the collections are added from 2.0.

# Difference between arraylist and vector ?

Both ArrayList and vector grows dynamically. The differences between arraylist and vector are :

1. Arraylist is not synchronized and vector is synchronized.
2. Vector is legacy collection introduced in 1.0 and Arraylist introduced in java 2.0.

Performance wise it is recommended to use arraylist rather than vector because by default vector is synchronized which reduces performance if only one thread accesses it.

# Define Linked List and its features with signature ?

Linked list is used for storing a collection of objects that allows efficient addition and removal of elements in the middle of the collection.

The main drawback with arrays is if we want to insert an element in the middle of the list we need to move each element to next position and insert the element. Similarly with remove if we want to remove an element we need to remove the element and move the list of elements.

But with linked list we can insert and delete in the middle of the list efficiently by just updating the neighboring node reference.

Linked list class is in java.util package.

Linked List class extends class extends AbstractSequentialList and I mplements List, Deque, Cloneable and Serializable.

# Explain about Sets ?

A set is a collection which does not allow duplicates. Set internally implements equals() method which doesn’t allow duplicates. Adding an duplicate element to a set would be ignored .Set interface is implemented in java.util.set package.Set interface does not have any additional methods . It has only collection methods. A set can contain at most one null value.

ArrayList is an ordered collection.In arraylists order remains same in which they are inserted. But coming to set it is an unordered collection.

public interface Set<E> extends Collection<E> {

}

Important operations that can be performed on set :

1. Adding an element to set.
2. Removing an element from set.
3. Check if an element exist in set.
4. Iterating through set.

# What is Hashtable and explain features of Hashtable?

Hashtable was available before collection framework.

When collection framework was started Hashtable extends Dictionary class and Map interface. Hashtable offers a convenient way of ***storing key/ value pairs***.

Hashtable ***does not allow nulls either keys or values***. Hashtable is ***synchronized***.

# Difference between HashMap and Hashtable?

|  |  |  |
| --- | --- | --- |
| **Difference** | **HashMap** | **Hashtable** |
| Synronization | HashMap is ***not synchronized***. | Hashtable is ***synchronized***. |
| Nulls | HashMap allows atmost ***one null key and any number of null values***. | Hashtable ***does not allow null values***. |
| Performance | Since HashMap is not synchronized its performance is ***faster than*** Hashtable. | Performance is ***slower*** when compared to HashMap. |
| Introduction | HashMap introduced starting from | Hashtable is even before collection |
|  | collection framework. | framework. |

1. **Difference between arraylist and linkedlist?**

|  |  |  |
| --- | --- | --- |
| **Difference** | **Arraylist** | **LinkedList** |
| Access | Implements RandomAccess interface we can ***search randomly*** all the elements  in the list. | It extends Abstract sequential List interface which provides  ***sequential access*** to elements. |
| Searching and retrieval of elements | ***Searching and retrieval*** of elements is  ***fast*** since arraylist provides random access. | ***Searching and retrieval*** of elements is ***slow*** because of  sequential access to elements. |
| Addition and removal of elements | ***Adding and removal of elements in random positions is slow***.For example if we want to add element to middle of the list we have to move the elements in the list and then we need to insert the element. Similarly for removing the element we need to follow the same  thing. | ***Adding and removal of elements in random positions is fast*** because there is no need of resizing the array just by updating the node structures with new addresses. |

# Difference between Comparator and Comparable in java?

|  |  |  |
| --- | --- | --- |
| **Sno** | **Comparator** | **Comparable** |
| 1. | Defined in ***java.util package*** | Defined in ***java.lang package.*** |
| 2. | Comparator interface is used when ***we want to compare two different instances*** | Comparable is used ***to compare itself with other instance***. |
| 3. | Comparator is used when we want ***custom sorting***.Ex : If we take employee class sorting by  employeeId is natural sorting. | Comparable is used for ***natural sorting*** of objects.Ex : If we take employee class sorting by ename and age we can say as custom sorting. |
| 4. | Should override int compare(T o1, T o2) method which takes two  instances. | Should override public int compareTo(T o) method which takes one instance. |
| 5. | For sorting objects we use collections.sort(list,new  Comparator); | For sorting objects we use collections.sort(list); |

1. **What is concurrent hashmap and its features ?**

Concurrent HashMap is implemented in *java.util.concurrent* package. Concurrent HashMap extends Abstract Map and implements concurrent Map. Concurrent HashMap is used in multi threaded environment.

]It is similar to Hashtable and synchronized version of hashmap but with minor differences. Concurrent HashMap *does not allow null keys and values*.

# Difference between Concurrent HashMap and Hashtable and collections.synchronizedHashMap?

Locking Mechansim :ConcurrentHashMap uses completely different hashing mechanism called *lock striping* which offers better concurrency and scalability.

The main advantage of this mechanism is better concurrency instead of synchronizing every method by using common lock which allows only one thread to access at a time, *it allows better concurrency by allowing multiple threads to access*.

ConcurrentModificationException :ConcurrentHashMap provides iterators which doesnot throw concurrent modification exception which allows only one thread to access iterator, *while synchronized map may throw concurrent modification exception*.

# Explain about fail fast iterators in java?

When iterator iterates over collection, collection should not be modified except by that iterator. Modification means collection cannot be modified by thread when other thread is iterating, if such modification happens a concurrent modification exception will be thrown.Such kind of iterators are fail fast iterators.

Ex : ArrayList,HashSet,HashMap. Almost all the iterators implemented in collections framework are fail fast.

# Explain about fail safe iterators in java?

Fail safe iterators are iterators which does not throw concurrent modification exception, when one thread modifies collection and other thread in the process of iterating the collection.

It does not throw concurrent modification exception because when other thread was iterating it does not modify original list but creates a copy of list with modified contents so that the iterator won’t know the modifications made to original list.

Ex : copyOnWriteArrayList