**Core Java**

Basics & Oops

1. **What is JAVA ?**

Java is a programming language and computing platform Java is an object oriented **programming language which follows the concepts of OOPS like Abstraction, polymorphism, inheritance and encapsulation.**

**2.What are the features of Java?**

* Simple
* Object Oriented
* Platform Independent
* Portable
* Robust
* Simple: It doesn’t have the concepts of operator overloading and explicit pointers. No need to remove the unreferenced objects because there is Automatic Garbage Collection in java.
* Object Oriented: It follows the concepts of OOPS like APIE(abstraction, polymorphism, inheritance and encapsulation).
* Platform Independent: The byte code makes java platform independent. As it can be run on any platform. (WORA)
* Portable: The byte code can be easily carried to any platform.
* Robust: Java is strong in terms of memory management, automatic garbage collection, no explicit pointers all these make Java Robust.
* Secured: It is secured because JVM has Classloader Its adds security by separating the package for the classes of the local file form those that are imported from network sources.

**3)What are the Different types of applications you can build with JAVA ?**

There are mainly four kind of applications we can build using java they are

**Standalone applications, enterprise applications, web applications, mobile applications**

**Standalone Applications:** or desktop applications is kind of applications that we need to install on every machine such as media player antivirus.

AWT and Swing are used in java for creating standalone applications

**Enterprise Applications:** this kind of applications are distributed in nature, such as banking applications etc.it has the advantage of high level security, load balancing, clustering

In java EJB is used for creating enterprise applications.

**Web Applications**: these are the applications that runs on the server side and generates dynamic web pages.in java we use servlets, JSPs, JSF, Struts technologies to develop web applications

**Mobile Applications**: An application that is created for mobile devices. currently Android and java ME are used for creating mobile applications

**4)What is path and classpath?**

Path and Classpath both are environment variables

Path is used by the operating system to locate the JDK binaries like “java”or “javac” commands and other executables which are used to run java programms. where as classpath is used by java compiler to find path so to find where the .class or .jar files are located.

**5)  Difference between JDK, JRE and JVM ?**

**JVM  :** It is an abstract machine which loads, verifies and executes the code, provides the runtime environment for the byte code to be executed. It directly comes in contact with OS but the code within the JVM never interacts with it which makes it more secured.

**JRE :** It is the implementation of the JVM. It has got all the libraries + all the files used by JVM at run time. It provides the runtime environment.

**JDK :** It consists of JRE and  utilities tools like javac , java for compilation and execution of java and bytecode respectively. In addition it also consists of development tools like javadoc e.t.c.

**6)Explain is the Java memory Model?**

Java memory model mainly consists of 5 areas.

**Class Area**: It consists of the static variables.

**Heap:** all The objects created are stored in heap.

**Stack:** It holds the primitive variables, partial results and local variables.

**Program Counter:** It consists of the address of the current instruction being executed.

**Native Methods:** it consists of the methods of other programming languages like C.

**7)Different data types in Java?**

              Data types represents different values to be in the variable. In java there are two types of data types.

                 1)  Primitive (Boolean, character, byte, short, int, long, double, float)

                 2)  Non-Primitive (String, Array)

**8)  Explain the design of Java class?**

              In java we design program using the template like

1. Package Name (domain Name. company Name. Project Name. functionality)
2. Import statements
3. Class Declaration (Access Modifier  classkeyword  ClassName)
4. Properties or data members (Access Modifier   data Types   variable Name)
5. Method Declaration (Access Modifier returnType methodName())
6. Comments

**9)Different types of operators?**

* Arithmetic Operator(\* / % + -)
* Assignment Operator(= += -= \*= /= %= &= ^= |= <<= >>= >>>=)
* Relational Operator(< > <= >= instanceof == !=)
* Bitwise Operator(& ^ |)
* Logical Operator(&& ||)
* Unary Operator(expr++ expr-- ++expr --expr +expr -expr ~ !)
* Ternary operator(? :)

10) **Top 5 operators ?**

These are the operators we use many times while developing applications

1) .(dot) operator

2) = operator

3) == operator

4) && and || operators

5) + operator

**11) Explain how each control statement works ?**

**(if-else, for, while, do-while, ternary, switch, break and continue)**

 These control statements can be classified into three groups: Decision making statements, repetition statements and branching statements.

**if:**

The Java if statement is used to test the condition. It checks boolean condition: true or false. There are various types of if statement in java.

* if statement
* if-else statement
* nested if statement
* if-else-if ladder

if(condition1){

//code to be executed if condition1 is true

}else if(condition2){

//code to be executed if condition2 is true

}

else if(condition3){

//code to be executed if condition3 is true

}

...

else{

//code to be executed if all the conditions are false

}

**Switch:** The Java switch statement executes one statement from multiple conditions.

**switch**(expression){

**case** value1:

 //code to be executed;

**break**;  //optional

**case** value2:

 //code to be executed;

**break**;  //optional

......

**default**:

code to be executed **if** all cases are not matched;

}

The java switch statement is fall-through. It means it executes all statement after first match if break statement is not used with switch cases.

**For Loop:** The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

* Simple For Loop

**for**(initialization;condition;incr/decr){

//code to be executed

}

**For-each or Enhanced For Loop:** The for-each loop is used to traverse array or collection in java

It works on elements basis not index. It returns element one by one in the defined variable

**for**(Type var:array){

//code to be executed

}

* Labeled For Loop: We can have name of each for loop. To do so, we use label before the for loop. It is useful if we have nested for loop

labelname:

**for**(initialization;condition;incr/decr){

//code to be executed

}

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

**for**(;;){

//code to be executed

}

**While Loop:**

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

**while**(condition){

//code to be executed

}

If you pass **true** in the while loop, it will be infinitive while loop.

**while**(**true**){

//code to be executed

}

**Do-While Loop:**The Java do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop. The Java do-while loop is executed at least once because condition is checked after loop body.

**do**{

//code to be executed

}**while**(condition);

**Break Statement:** The Java break is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

**Continue Statement:** The Java continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

**Ternary Operator:**

**class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** a=2;

**int** b=5;

**int** min=(a<b)?a:b; //a<b is true so it gives a from a:b if a<b is false it gives b

System.out.println(min);

}} // Output:2

––

**12)What is class and object?**

Class: Class is a template which represents state and behaviour of an object. In java state is implemented through properties and behaviour is implemented through methods. class is collection if objects.

     Object: Object is instance of a class.

**13) What are different OOPS concepts?**

o   Abstraction

o   Polymorphism

o   Encapsulation

o   Inheritance

**What is abstraction?**

Abstraction: Abstraction is the **concept** of hiding the implementation and showing only the essential features of the object. Abstraction mainly comes into picture for future flexibility when the developer is not sure of the implementation.

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated. A method that is declared as abstract and does not have implementation is known as abstract method.

There are two ways to achieve abstraction:

1. Abstract class.
2. Interface.

If we use abstract class we can achieve 0-100% abstraction where as if we use interface we can achieve 100% abstraction.

Abstraction mainly comes into picture for future flexibility when the developer is not sure of the implementation

**What is Encapsulation?**

Encapsulation is one of the object orientedconcept which enforces protecting variables, functions from outside of the class by making them private using private keyword and providing access only through getters and setters . using encapsulation, we can provide security.

**What is the difference between abstraction and encapsulation?**

Abstraction and Encapsulation are two important [Object oriented programming concept](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html)s and they are completely different to each other.

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

**Encapsulation in java** is a process of wrapping data and code(methods) which is acting on data together into a single unit.

First difference between Abstraction and Encapsulation is that, Abstraction is implemented in Java using [interface](http://javarevisited.blogspot.sg/2012/04/10-points-on-interface-in-java-with.html) and abstract class while Encapsulation is implemented using [private](http://javarevisited.blogspot.sg/2012/03/private-in-java-why-should-you-always.html) keyword.

Design principles "[programming for interface than implementation](http://javarevisited.blogspot.sg/2012/06/20-design-pattern-and-software-design.html)" is based on abstraction and "encapsulate whatever changes" is based upon Encapsulation.

**What is inheritance?**

Inheritance is the **concept** of acquisition of properties and behavior from one class to another. It is mainly used for code reusability. It is an IS-A relationship. The extends keyword is used to extend once class to another. A class that is inherited is called a super class. The new class is called a subclass.

**What is Polymorphism?**

Polymorphism is the concept of an object’s ability to take on many forms. There are two types of polymorphism:

     1.Compile time polymorphism

                                   2.Runtime polymorphism.

The polymorphism which occur during method overloading is called compile time polymorphism because the method calling will be resolved at compile time

The polymorphism which occur during method overriding is called runtime polymorphism because the method calling will be resolved at runtime depending upon the actual object and not the type of variable.

**What is method overloading and overriding?**

Method overloading is the **concept** of a class having multiple methods of same name but vary with type and number of arguments. It enhances the readability. It is performed within a class. method overloading is resolved at compile time

Method overriding is the **concept** of implementing a method in the sub class which is already provided in the super class. Both classes should compulsorily be in an IS-A relation.  The subclass method must have the same method name and the parameters. method overriding is resolved at runtime.  
  
 You cannot override private , static and final methods from a superclass.

**What is static in JAVA ?**

Static is a keyword in java which you can apply towards variables, methods and blocks and inner classes

If you apply towards variables the variable becomes class level variable and if you apply towards methods we need not to create the instance of the class to call the method and if we apply towards blocks these blocks will be executed before the constructor.

this and super keywords can not be used in static methods.

**27)What is final in java?**

final is a keyword in java which you can apply towards variables, methods and classes

If we apply towards variables the value can not be changed. and if we apply towards methods we can not override the method in the child class. if you apply towards classes we can not extend the class.

28) **What is final, finally, finalize ?**

**final:** final is a keyword in java which you can apply towards variables, methods and classes

If we apply towards variables the value can not be changed. and if we apply towards methods we can not override the method in the child class. if you apply towards classes we can not extend the class.

**finally:** finally is another Java keyword which is used in Exception handling concept along with try, catch, [throw and throws](http://javarevisited.blogspot.sg/2012/02/difference-between-throw-and-throws-in.html).

We write finally block after the try or catch blocks. finally block have an advantage that no matter what the code inside the block will be executed whether the exception occurred or not, handled or not.so we write important code like close system resource e.g. [InputStream](http://javarevisited.blogspot.sg/2012/08/convert-inputstream-to-string-java-example-tutorial.html) or OutputStream and closing  network connection, database connection

**finalize:** finalize() is a special method in Java which is called by [Garbage Collector](http://javarevisited.blogspot.sg/2011/04/garbage-collection-in-java.html) before object is eligible for garbage collection. This is the last chance for an object to perform any cleanup activity.

**What is static and dynamic binding?**

The **concept** of Establishing a connection between method body and method call is known as binding. There are two types of binding.

* Static Binding
* Dynamic binding

Static binding: The **concept** of determining the object at the compile time then it is known as early binding or static binding. When a class has final, static or private method is also known as static binding.

Dynamic binding: The **concept** of determining the type of the object at the run time it is known as dynamic binding. Overriding is a perfect example of dynamic binding as in overriding both parent and child classes have same method. Thus while calling the overridden method, the compiler gets confused between parent and child class method. this will be resolved by JVM at runtime.

Abstract class vs interface ?

|  |  |  |
| --- | --- | --- |
|  | **abstract Classes** | **Interfaces** |
| 1 | abstract class can extend only one class or one abstract class at a time | interface can extend any number of interfaces at a time |
| 2 | abstract  class  can extend from a class or from an abstract class | interface can extend only from an interface |
| 3 | abstract  class  can  have  both  abstract and concrete methods | interface can  have only abstract methods |
| 4 | A class can extend only one abstract class | A class can implement any number of interfaces |
| 5 | In abstract class keyword ‘abstract’ is mandatory to declare a method as an abstract | In an interface keyword ‘abstract’ is optional to declare a method as an abstract |
| 6 | abstract  class can have  protected , public and public abstract methods | Interface can have only public abstract methods i.e. by default |
| 7 | abstract class can have  static, final  or static final  variable with any access specifier | interface  can  have only static final (constant) variable i.e. by default |

**Which one you choose between abstract and interface?**

Use an abstract class if you have some functionality that you want it's subclasses to have. For instance, if you have a set of functions that you want all of the base abstract class's subclasses to have.

Use an interface if you just want a general contract on behavior/functionality. If you have a function or object that you want to take in a set of different objects, use an interface. Then you can change out the object that is passed in, without changing the method or object that is taking it.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Abstract classes are used to group a number of concrete classes under one entity.  For example, take the abstract class Animal. Animal is not something concrete. it's a family of, well, animals. but they all share certain aspectes, for example, each has a speak() option (well, except fish and sort). but each one implements it differently. this way you can override just the methods which are not the same, for example sleep() or breath() are common (again, fish are differnet :) ).  Interfaces on the other hand are more direct definition of an 'action'. That's why most (if not all) the interfaces in Java ends with 'able' (Comprable, Serializable...) By implementing the interface, you're telling other programmers or who ever uses your code that this class can do this and this. A dog, for example, is not, Animable.  **Why do you create abstract classes in application development?**  For flexibility of the code in the future and also when the developer is not sure of the implementation. The real world abstract things can be considered to be abstract.  **Why do you create interfaces in application development?**  When multiple classes have similar functionality but vary in the implementation of that particular functionality then interfaces come into picture. It provide reusability, maintainability, and extensibility.  **What are different types of access modifiers?**  Public: Accessible everywhere.  Private: Accessible within the class.  Protected: Accessible within the package and outside the package only through inheritance.  Default: Accessible only within the package.  **What is call by value?**  Call by value is a mechanism where we pass the parameters to a method by calling that method.java supports only call by value.  Usually we pass two kinds of parameters primitive types or Object references  When we pass primitive type as parameters, value held in the variable that is passed as an argument is copied into the parameters that are defined in the method header. That is why changes made to the variable within the method had no effect on the variable that was passed.  when an object reference is passed to a method, the method gets a value of the object reference, and both the actual and the formal parameters refer to the same object, therefore within from the method the state of an object parameter can be changed.  **What is Has-A relation?**  If a class have an entity reference of other class ,then it is known as Aggregation. Aggregation represents HAS-A relationship. there is no specific keyword to implement HAS-A relationship but mostly we are depended upon “new” keyword.  Strings  **What is String in Java?**  In java, string is basically an object that represents sequence of  4characters , java provides String class to create and manipulate strings. String objects are immutable.  There are two ways to create String object:   1. By string literal 2. By new keyword   String objects are stored in Special memory area known as String constant pool  **What is String constant pool ?**   String Pool is a special area with **pool of strings** in [Java Heap Memory](http://www.journaldev.com/4098/java-heap-space-vs-stack-memory)**.** String Pool is possible only because [String is immutable in Java](http://www.journaldev.com/802/string-immutable-final-java)  When we use String literals to create a String, it first looks for String with same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference.  However using new operator, we force String class to create a new String object in heap space. We can use intern() method to put it into the pool or refer to other String object from string pool having same value.  **Why String is immutable ?**  Most of the data is represented in the form of strings in our application development .so there is a high chance that two Strings may point to the same value. If a string changes, it will affect the other string which is being referenced to the same pointer and there are chances that it might become unreferenced which gets removed by the automatic garbage collector. This is the reason why strings are made immutable.  http://www.java67.com/2014/01/why-string-class-has-made-immutable-or-final-java.html  **How do you make a class immutable ?**  By declaring a class final.  By declaring the properties of the class as final.  By not providing the setters.  **StringBuffer vs StringBuilder?**   |  |  |  | | --- | --- | --- | | **No.** | **StringBuffer** | **StringBuilder** | | 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. | | 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |   **What is** difference between **== vs equals**  == operator is used to used compare primitives along with the objects .equals method is used to check the equality of objects  If we use == operator to compare objects it checks the reference s of two Strings whether they are pointing to the same String or not based on this it returns true or false  If we use equals method it checks the content of String it returns true if the content is equal otherwise false.  **Top 5 methods in String class ?**  equals(), substring(), split(), length(),chatAt(),  Exception Handling  What is exception handling and how do you achieve it?  Exception handling is a problem that occurs during the execution of the program. It disrupts the normal flow of the program and the application terminates abnormally, which is not recommended so these exceptions are to be handled. It is achieved through try, catch and throw blocks.  **How do you make sure a code must be executed even if exception happens?**  By placing the code in the finally block. The finally block gets executed even the exception is handled or not.  **What code you normally write in finally block?**  Most important statements like closing connections and statements are written in finally block. It should always be followed by try or catch block. The finally block gets executed even the exception is handled or not.  **What are checked vs unchecked exceptions?**  Checked exceptions are exceptions that occur at the compile time. The extend only the throwable class. Example: File not found exception.  Unchecked exceptions are exceptions that occur at the run time. For example: - array out of bounds exception.  **How do you create custom exceptions?**  If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need.Throw keyword is used to throw custom exceptions. By Using the throw keyword, we can explicitly throw exceptions either checked or unchecked exceptions.  **How does exception propagation works?**  An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method, if not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.  **What are inner classes ?**  **Java inner class** or nested class is a class i.e. declared inside the class or interface.  We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.  Additionally, it can access all the members of outer class including private data members and methods.  **What is Anonymous class ?**  A class that have no name is known as anonymous inner class in java. It should be used if you have to override method of class or interface. Java Anonymous inner class can be created by two ways:  Class (may be abstract or concrete).  Interface  **ArrayList vs LinkedList**  ArrayList and LinkedList implements List Interface. If we want to represent group of objects as single unit where insertion order is maintained and duplicates are allowed we can use anything either ArraryList or LinkedList but there are significant differences between them like ArrayList is index based and LinkedList is node based.  ArraryList:-  • Internally it uses dynamic array, if any element is removed from an array all the bits are shifted in memory. So, manipulation is slow.  • ArrayList is preferred if our operation is based on searching an element as ArrayList implements RandomAccess.(either first element or last element it finds with same speed).  LinkedList:-  • LinkedList uses doubly linked list to store elements. If any element is removed from the list then no bit shifting is required in memory. So, manipulation is fast. So, LinkedList is preferred if our operation is for adding or removing basically I could say for maupulation.  **List vs Set:-**  List and set are interfaces which extends collection interface.  List:-  If we want to represent group of objects as single unit where insertion order is maintained and duplicates are allowed we use list interace  ArrarList, LinkedList, Vector implements List Interface.  Set:-  If we want to represent group of objects as single unit where insertion order is not maintained and duplicates are not allowed we use Set Interface. But there are few implementatons in set which maintains the insertion order is LinkedHashSet.  We know that List allow duplicates while Set doesn’t allow if you are trying to insert a duplicate element then old value is replaced by new value.  HashSet, LinkedHashSet, TreeSet implements Set interface.  **Which one you prefer beween ArrayList vs LinkedList?**  LinkedList allows constant-time insertions or removals using iterators but only sequential access of elements. In other words you can walk through elements using forward or backward directions. In order to find an element it takes a lot of time it is almost proportional to size of elements in the list.  ArraryList:-  On other hand in ArraryList we can find any element with same speed either first or last element we can find it with same speed because ArrayList implements RandomAccess. While in Insertion or Removal of elements bits should be shifted in memory, and also If you add more elements than capacity, a new array list is created and old array is copied and new elements will be added.  **How do you sort collection of elements in JAVA ?**  Collections.sort can be called with a custom comparator. And that comparator can be implemented to allow sorting in different sort orders.  **What is Collections class in JAVA ?**  Collections class provides static methods for sorting the elements of collection.If collection elements are of Set type, we can use TreeSet.But We cannot sort the elements of List.Collections class provides methods for sorting the elements of List type elements.  **HashTable vs HashMap?**  HashTable and HashMap implements Map interfaces. If we want to represent group of objects as K-V based collection we use HashTable and HashMap.  HashMap:- HashMap is non-synchronized and not-thread safe. So, two threads can call a method directly. It is fast. It allow one null key and multiple null values.  Hashtable:-  It is synchronized and thread safe. So, two threads cannot call a method directly. So, it is slow. It doesn’t allow any null keys or values.  **What is HashMap?**  HashMap implements Map interface. If we want to represent group of objects as k-V based collection then we use HashMap. It contains only unique elements. It may have one null key and multiple null values.  **How does HashMap Works?**  HashMap implements Map interface, and it is K-V based collection. Basically, HashMap deals with put() and get() methods.  Put()  • First it checks whether key is null or not. If key is null putForNullKey() method is called and hashcode of null key is always zero. So it enters zeroth index in entry Map.  • If key is not null hashcode of key is calculated based on hashCode() method.  • Once hashcode is calculated based on the index it entry to map entry.  • After getting index, if there is no key present for that particular index a bucket is formed.  • If bucket is already present and if the current key is equal to already present key in bucket then it replaces the old value with new value, Otherwise it appends the K-V pair at end of linked list.  get()  • First checks whether key is null or not if key is null then it calls getForNullKey() method.  • Key is not null hashcode of the key is calculated.  • indexFor() method is called for calculating the index of the key. Based on index it enter into map entry.  • Now it will iterate to list of key present and if there is any equal key present in linkedList when it returns the value of it. If no matching found it returns null.  **What is HashMap collison?**  Hash Map collision comes into picture if a bucket is already present in map entry at a particular index, and if any key returns same hashcode for that particular index then hashmap injection comes into picture.  **Why do you think Strings makes appropriate keys for HashMap ?**  An immutable key will always keep the same hashCode() value, and the hashing  function will find the correct bucket ( = index in the hashMap's array) again.  **What kind of classes are good for HashMap keys ?**  Immutable classes  **What is Iterator vs ListIterator?**  Iterator:-  1) Iterator is used for traversing List and Set both.  2) We can traverse in only forward direction using Iterator.  3) We cannot obtain indexes while using Iterator.  4) We cannot add element to collection while traversing it using Iterator, it throws ConcurrentModificationException when you try to do it.  5) We cannot replace the existing element value when using Iterator.  List Iterator:-  • We can use ListIterator to traverse List only, we cannot traverse Setusing ListIterator.  • Using ListIterator, we can traverse a List in both the directions (forward and Backward).  • We can obtain indexes at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose.  • We can add element at any point of time while traversing a list using ListIterator.  • By using set(E e) method of ListIterator we can replace the last element returned by next() or previous() methods.  **Comparable vs Comparator interfaces**  Comparable and Comparator are interfaces in java. In which we use them to sort collection of object.  Comparable:-  • Comparable we can sort the collection on basis of single element. So, Comparable provides single sorting sequence.  • Comparable provides compareTo() method to sort elements.  Comparator:-  • Comparator we sort the collection on basis of multiple elements, So, comparator provides multiple sorting sequence.  • Comparator provides compare() to sort elements.  **Treeset vs TreeMap**  1) Major difference between TreeSet and TreeMap is that TreeSet implements Set interface while TreeMap implements Map interface in Java.  2) TreeSet stores only one object while TreeMap uses two objects called key and Value. Objects in TreeSet are sorted while keys in TreeMap remain in sorted Order.  **Why do you need override Hash code and equals**  Every object is placed in Hash bucket depending on the hashcode they have. It is not necessary that every different object must have different hashcode. hashcode is used to narrow the search result. When we try to insert any key in HashMap first it checks whether any other object present with same hashcode and if yes then it checks for the equals() method. If two objects are same then HashMap will not add that key instead it will replace the old value by new one. |

Exception Handling

**What is exception handling and how do you achieve it?**

Exception is an abnormal event that occurs during the execution of the program. It disrupts the normal flow of the program and the application terminates abnormally, which is not recommended so these exceptions are to be handled. It is achieved through try, catch and throw blocks.

**How do you make sure a code must be executed even if exception happens?**

By placing the code in the finally block.

30)What code you normally write in finally block?

Most important statements like closing connections and statements are written in finally block. It should always be followed by try or catch block. The finally block gets executed even the exception is handled or not.

31)What are checked vs unchecked exceptions?

Checked exceptions are exceptions that occur at the compile time. The extend only the throwable class. Example: File not found exception.

Unchecked exceptions are exceptions that occur at the run time. For example: - array out of bounds exception.

32)How do you create custom exceptions?

Using the throw keyword, we can explicitly throw exceptions either checked or unchecked exceptions. It is mainly used to throw custom exceptions.

33)How does exception propagation works?

An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method, if not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.

34)exception vs error?

|  |  |
| --- | --- |
| Errors | Exceptions |
| Errors in java are of type java.lang.Error. | Exceptions in java are of type java.lang.Exception. |
| All errors in java are unchecked type. | Exceptions include both checked as well as unchecked type. |
| Errors happen at run time. They will not be known to compiler. | Checked exceptions are known to compiler where as unchecked exceptions are not known to compiler because they occur at run time. |
| It is impossible to recover from errors. | You can recover from exceptions by handling them through try-catch blocks. |
| Errors are mostly caused by the environment in which application is running. | Exceptions are mainly caused by the application itself. |
| Examples : java.lang.StackOverflowError, java.lang.OutOfMemoryError | Examples : Checked Exceptions : SQLException, IOException Unchecked Exceptions : ArrayIndexOutOfBoundException, ClassCastException, NullPointerException |

### What is try-with-resources in java?

One of the Java 7 features is try-with-resources statement for automatic resource management. Before Java 7, there was no auto resource management and we should explicitly close the resource. Usually, it was done in the finally block of a try-catch statement. This approach used to cause memory leaks when we forgot to close the resource.From Java 7, we can create resources inside try block and use it. Java takes care of closing it as soon as try-catch block gets finished.

### What is multi-catch block in java?

Java 7 one of the improvement was multi-catch block where we can catch multiple exceptions in a single catch block. This makes are code shorter and cleaner when every catch block has similar code.If a catch block handles multiple exception, you can separate them using a pipe (|) and in this case exception parameter (ex) is final, so you can’t change it.

### Can we have try without catch block?

Yes, we can have try-finally statement and hence avoiding catch block.

What are inner classes ?

**Java inner class** or nested class is a class i.e. declared inside the class or interface.

We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.

Additionally, it can access all the members of outer class including private data members and methods.

What is Anonymous class ?

A class that have no name is known as anonymous inner class in java. It should be used if you have to override method of class or interface. Java Anonymous inner class can be created by two ways:

Class (may be abstract or concrete).

Interface

Multi-Threading

**What is Multi-Threading in java?**

Multithreading in java is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing ,In Multithreading we can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel so that we can utilize resources properly and improve the performance by reducing the processing time.

**How do you create Threads?**

There are two ways to create a thread in Java:  
a) By extending Thread class.

You will need to override run**( )** method available in Thread classOnce Thread object is created, you can start it by calling start**()** method, which executes a call to run( ) method.

b) By implementing Runnable interface.

As a first step, you need to implement a run() method provided by a Runnable interface. As a second step, you will instantiate a Thread object using the  Thread  constructor Once a Thread object is created, you can start it by calling start**()** method, which executes a call to run( ) method

**How do you synchronize your code ?**

Synchronization in java is the capability to control the access of multiple threads to any shared resource. The synchronization is mainly used to prevent thread interference. And prevent consistency problem. There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
   1. Synchronized method.
   2. Synchronized block.
   3. static synchronization.
2. Cooperation (Inter-thread communication in java)

**What is volatile ?**

volatile keyword is used to make any variable volatile in Java environment

it indicate that a variable's value will be modified by different threads. Java volatile keyword cannot be used with method or class and it can only be used with variable

Declaring a volatile Java variable means:

The value of this variable will never be cached thread-locally: all reads and writes will go straight to "main memory";

Any variable which is shared between multiple threads should be made variable, in order to ensure that all thread must see the latest value of the volatile variable.  
A signal to compiler and JIT to ensure that compiler does not change ordering or volatile variable and moves them out of synchronized context.  
You want to save the cost of synchronization as volatile variables are less expensive than synchronization.  
  
Read more: <http://www.java67.com/2012/08/what-is-volatile-variable-in-java-when.html#ixzz4dK6rKV2q>

**What is Race Condition ?**

Race condition occurs in a multi-threaded environment when more than one thread try to access a shared resource (modify, write) at the same time. Because the thread scheduling algorithm can swap between threads at any time, you don't know the order in which the threads will attempt to access the shared data. As the order in which they access and modify it is unknown, the output is ambiguous Since multiple threads try to race each other to finish executing a method thus the name **race condition.** To avoid it synchronize the code using a synchronized block.

**What is dead lock ?**

Deadlock in java is a part of multithreading. Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Consider a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, In such case, they will end up waiting forever. the condition is called deadlock.

**What is ThreadLocal ?**

The ThreadLocal class in Java enables you to create variables that can only be read and written by the same thread a single instance of ThreadLocal can store different values for each thread independently. Therefore, the value stored in a ThreadLocal instance is specific (local) to the current running Thread if different threads execute the same code which accesses a ThreadLococal, each thread will see only its own ThreadLocal instance. Even if two different threads set different values on the same ThreadLocal object, they cannot see each other's values.

**What is ThreadPool ?**

A thread pool is a group of threads initially created that waits for jobs and executes them. We can use ThreadPoolExecutor to create thread pool in java Thread Pools are useful when you need to limit the number of threads running in your application at the same time. There is a performance overhead associated with starting a new thread, and each thread is also allocated some memory for its stack etc.

Instead of starting a new thread for every task to execute concurrently, the task can be passed to a thread pool. As soon as the pool has any idle threads the task is assigned to one of them and executed. Internally the tasks are inserted into a [Blocking Queue](http://tutorials.jenkov.com/java-concurrency/blocking-queues.html) which the threads in the pool are dequeuing from. When a new task is inserted into the queue one of the idle threads will dequeue it successfully and execute it. The rest of the idle threads in the pool will be blocked waiting to dequeue tasks.

**How do you use Executor framework ?**

### [Executors framework](http://docs.oracle.com/javase/tutorial/essential/concurrency/executors.html) (java.util.concurrent.Executor), released with the JDK 5 in package java.util.concurrent is used to run the Runnable objects without creating new threads every time and mostly re-using the already created threads.and to decouple command submissions from command execution.

### What is starvation ?

**Starvation** occurs when a thread is continually denied access to resources and as a result it is unable to make progress. This usually happens when greedy threads consume shared resources for long periods of time. When this happens for extended periods of time, the thread not getting enough CPU time or access to the resource will not be able to make enough progress leading to [**thread starvation**](http://docs.oracle.com/javase/tutorial/essential/concurrency/starvelive.html). One of the likely causes of thread starvation is incorrect thread priorities among different threads or thread groups.and use of non-terminating loops (infinite loops) or waiting excessive amount of time on specific resources

.

**Synchronized methods vs Synchronized blocks ?**

Synchronized block and synchronized methods are two ways to use [synchronized keyword in Java](http://javarevisited.blogspot.com/2011/04/synchronization-in-java-synchronized.html)  
The key difference is if you declare a method to be synchronized, then the entire scope of the method becomes synchronized; if you use the synchronized block, however, then you can surround just the "critical section" of the method in the synchronized block, while leaving the rest of the method out of the block.

If the entire method is part of the critical section, then there effectively is no difference. If that is not the case, then you should use a synchronized block around just the critical section

Though both block and method can be used to provide highest degree of synchronization in Java, use of synchronized block over method is considered as better

Read more: <http://www.java67.com/2013/01/difference-between-synchronized-block-vs-method-java-example.html#ixzz4dLW5bwEK>

**What is serialization and externalization ?**

 Externalizable and Serializable interface both are interfaces Serializable does not contain any method but Externalizable interface contains two methods writeExternal() and readExternal()

**Serialization** is the process of converting an object into a sequence of bits so that it can be persisted on a storage medium (such as a file, or a memory buffer) or transmitted across a network connection link. The reverse operation of serialization is called *deserialization*

Externalization allows you to customize how serialization is done. By implementing externalization you are controlling what gets serialized ( and what doesnot ) **Externalization** is same as Sterilization except that WriteObject() and ReadObject() method are called by JVM during sterilization an desterilization of object. One thing you can do with Externalization is that you can store extra information into object like STATIC variables and transient variables or you can add more information if you have any business need  
  
  
Read more: <http://www.techartifact.com/blogs/2009/06/serialization-vs-externalization.html#ixzz4dLeBGOWy>

**How do you avoid a variable particiapting in serialization ?**In One-word **transient keyword** is used in serialization process to prevent any variable from being serialized, so if you have any field which is not making sense to serialize, you can simply declare that as transient and it won't be serialized.  
  
Read more: <http://javarevisited.blogspot.com/2011/09/transient-keyword-variable-in-java.html#ixzz4dLids5rG>

### What is Marker interface?

A marker interface is an empty interface without any method but used to force some functionality in implementing classes by Java. Some of the well known marker interfaces are Serializable and Cloneable.

### What is Enum in Java?

Enum was introduced in Java 1.5 as a new type whose fields consists of fixed set of constants. For example, in Java we can create Direction as enum with fixed fields as EAST, WEST, NORTH, SOUTH. enum is the keyword to create an enum type and similar to class. Enum constants are implicitly static and final.