**Core java Interview Questions**

1. [How to create a immutable object in Java? Count all benefits?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques1)
   * **Immutable objects** are instances whose state doesn’t change after it has been initialized.

For example, String is an immutable class and once instantiated its value never changes.

* + Benefits:
    1. It is thread safe
    2. An immutable class is good for caching purposes because you don’t have to worry about the value changes
  + Process to create immutable object:
    1. Declare class as Final [ so it cannot be extended]
    2. Make all field as private [ to restrict direct access]
    3. Remove setter methods if any [ as state of immutable object cannot get changed]
    4. Make all mutable fields final [ so value can be assigned only once]
    5. Initialize all fields with deep copy [ i.e. instead of copying reference use deep copy]
    6. In getter method return copy of object rather than directly returning object

[ if we return object directly, the value can be overridden since reference would have been pointing to same memory location]

1. [What is the use of the finally block? Is finally block in Java guaranteed to be called? When finally block is NOT called?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques3)
   * Finally block is used for operations like closing resources, closing files, freeing up threads etc. since it is called in all the cases.
   * Only in case of some scenarios the finally block cannot be called\_
     1. If JVM crashes as it runs out of memory, and complete java process get killed from system
     2. If System.exit(int) get called before finally block get called.
2. [Why main() in java is declared as public static void main?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques6)
   * Public: to make it accessible to external world
   * Static: as main method is an entry point, it can not be instance method (since instance will not be available)
   * void: will not return any value
   * main: name of the method (can be anything)
3. [What is the difference between creating String as new() and literal?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques7)
   * **When we create a**String**object using the**new()**operator, it always creates a new object in heap memory.**
   * **On the other hand, if we create an object using**String**literal syntax e.g. “Baeldung”, it may return an existing object from the String pool, if it already exists.** Otherwise, it will create a new String object and put in the string pool for future re-use.
   * At a high level, both are the String objects, but the main difference comes from the point that new() operator always creates a new String object. Also, when we create a String using literal – it is interned.
4. [How does subString() inside String works?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques8)
   * subString is an overloaded method present in String class
   * subString(int startIndex); or subString(int startIndex,int endIndex); can throw IndexOutOfBound exception
   * from Java7 onwords, it uses character Sequence to create new String object in heap or String pool and can be referred in by new String
5. [Difference between interfaces and abstract classes?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques10)

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) as Abstract class is a class multiple inheritance is not allowed | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables** |
| 4) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |

1. [When do you override hashCode and equals()?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques11)
   * When we want to change the logic of equals we can override equals method of Object class.
   * The default behavior of equals method is it return true if hashCode of two objects are equal, hence as per contract between equals and hashCode method, we must override hashCode method whenever we override equals method.
2. [Explain abstraction and encapsulation? How are they related?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#abstraction-encalsulation)
   * Abstraction and Encapsulation are pillars of Object oriented programming
   * Where Abstraction means showing only essential details where as Encapsulation means binding data and related properties together.
   * For example; in a class Car
     1. There can be methods like Car can goForward,gobackword,stop,horn,takeLeft etc.
     2. But for taking turns, internally we are changing value of direction,
     3. So as a user, goForword(int value) method will get called but internally it is changing the variable direction.
     4. Since as end user the essential method is goForward() will be called, but since since because of encapsulation the value of direction variable getting changed
3. [How StringBuffer save the memory?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#stringbuffer-save-memory)
   * String buffer is a mutable version of String class
   * As String class is immutable whenever we create any new object using String, new Object reference is added in the memory. Which increase memory consumption.
   * On the other hand, if we use StringBuffer it is immutable and thread safe, also there are methods available in StringBuffer class to modify the String values , so instead of creating new object it works on same object hence it saves memory.
4. [Explain transient and volatile keywords in java?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#transient-volatile)
   * A volatile keyword is used in a multithreading environment where two threads reading and writing the same variable simultaneously.
   * The volatile keyword flushes the changes directly to the main memory instead of the CPU cache.
   * On the other hand, the transient keyword is used during serialization.
   * Fields that are marked as transient cannot be part of the serialization and deserialization.
   * If we don't want to save the value of any variable then we use transient keyword with that variable.

| **Sr. No.** | **Key** | **Volatile** | **Transient** |
| --- | --- | --- | --- |
| 1 | Basic | Volatile keyword is used to flush changes directly to the main memory | The transient keyword is used to exclude variable during serialization |
| 2. | Default value | Volatile are not initialized with a default value. | During deserialization, transient variables are initialized with a default value |
| 3 | Static | Volatile can be used with a static variable. | Transient cannot be used with the static keyword |
| 4 | Final | Volatile can be used with the final keyword | Transient can not be used with the final keyword |

1. [Difference between Iterator and ListIterator?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#iterator-vs-listiterator)
   * Iterator can traverse only in forward direction whereas ListIterator traverses both in forward and backward directions.
     1. Iterator has only next() method
     2. whereas listIterator has next() and previous() methods
   * ListIterator can help to replace an element whereas Iterator cannot.
     1. Set() method present in ListIterator is used to set value while iterating
   * We can use iterator with Map, List and Set, whereas listIterator can only be used with List
   * We can find index using nextIndex() and previousIndex() methods in ListIterator, these methods are not available in iterator
   * Methods in Iterator interface : next(), remove() and hasNext() etc.
   * Methods in ListIterator interface : next(), previous(), hasNext(), hasPrevious(), add(E e) etc.
2. [Why finalize() method should be avoided?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#why-avoid-finalize)
   * Finalize method is called by the garbage collector on an object when it determines that there are no more references to the object.
   * A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.
   * But execution of finalize() method is not guaranteed.
   * Also, exception thrown by finalize() methods will not be caught by GC
   * Finalize method doesn’t support method chaining, i.e. if you want to use finalize() method, you must call finalize method of parent class in child class finalize() method.

Hence..

1. Deep copy and shallow copy?
   * Shallow copy is copying reference, in this case two or more variables may refers to a same instance
   * Deep copy is copying object data into new object, such that it will have separate memory location with same data.

1. Explain all access modifiers? (Done earlier)
2. What is garbage collection? Can we enforce it?
   * By calling finalize() method
3. What is native keyword?
   * The native keyword is applied to a method to indicate that the method is implemented in native code using JNI (Java Native Interface).
   * native is a modifier applicable **only for methods** and we can’t apply it anywhere else. The methods which are implemented in C, C++ are called as native methods or foreign methods.
4. What if the difference between && and &??
   * The key difference between && and & operators is that && supports short-circuit evaluations while & operator does not.
   * Another difference is that && will evaluate the expression exp1, and immediately return a false value if exp1 is false. While & operator always evaluates both expressions (exp1 and exp2) before retiring an answer

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Basis** | **& Operator** | **&& Operator** |
| **1** | **Operator** | It is a bitwise AND operator. | It is a logical AND operator. |
| **2** | **Evaluation** | It evaluates both the left and right side of the given expression. | It only evaluates the left sides of the given expression. |
| **3** | **Operates on** | It operates on Boolean data types as well as on bits. | It operates only on Boolean datatype. |
| **4** | **Uses** | Used to check logical condition and also used to mask off certain bits such as parity bits. | Used only to check the logical conditions. |
| **5** | **Example** | z = x & y | if (y > 1 && y > x) |

1. [How to create an instance of any class without using new keyword](https://howtodoinjava.com/puzzles/how-to-create-an-instance-of-any-class-without-using-new-keyword/)

Class ref = Class.forName("DemoClass");

DemoClass obj = (DemoClass) ref.newInstance();

Class.forName() loads the class in memory. To create an instance of this class, we need to use newInstance()

1. Is String thread-safe in Java?
   * Yes, As String is immutable it is thread safe
   * In multithread environment, even if multiple threads are pointing to single object its value cannot be changed hence it is thread safe.
   * As String is a widely used class, in JVM it separate memory space is allocated for String called String pool where actual string literal will be saved

1. How do you count the number of occurrences of each character in a string?

public class CountOccurences

{

  public static void main(String args[])

  {

  String input = "aaaabbAAAAcccddd";

  char search = 'a';             // Character to search is 'a'.

  Map<Character,Integer> hash = new HashMap<Character,Integer>();

  for(int i=0;i<input.length();i++)

  {

      if(hash.containsKey(input.charAt(i)))

      hash.put(input.charAt(i), hash.get(input.charAt(i))+1);

      else

      hash.put(input.charAt(i), 1);

  }

  int result = hash.get(search);

  System.out.println("The Character '"+search+"' appears "+result+" times.");

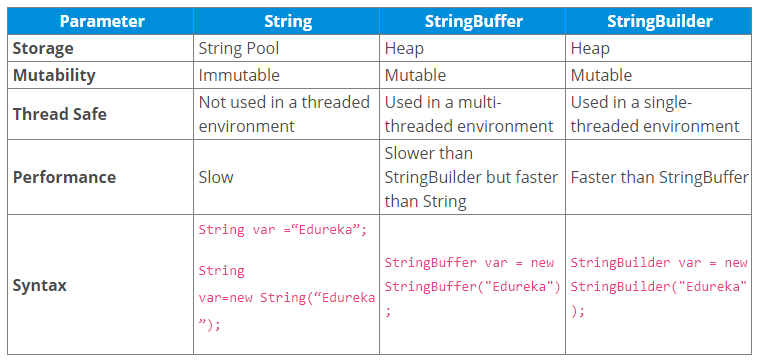
  }

}

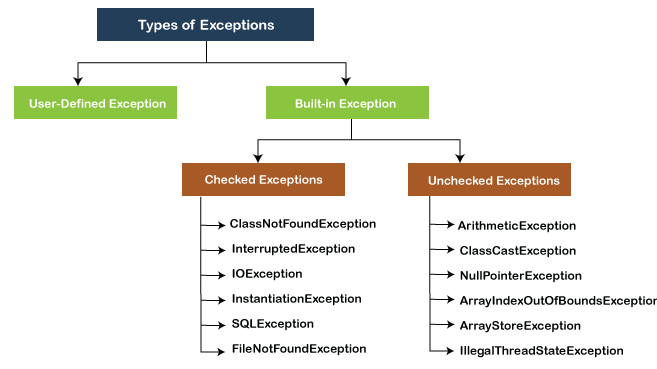
1. Write a java program to reverse a string?

**class** InvertString  
{  
  **public** **static** **void** main(String args[])  
  {  
     StringBuffer a = **new** StringBuffer("Java programming is fun");  
     System.out.println(a.reverse());  
  }  
}

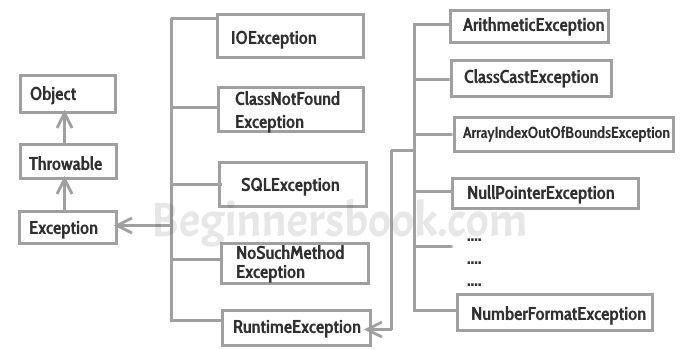
1. Difference between String, StringBuffer and StringBuilder?



1. Memory leak issue in String class :
2. Why are strings immutable?
   * The string is Immutable in Java because String objects are cached in the String pool.
   * Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client.
   * For example, if one client changes the value of the String "Test" to "TEST", all other clients will also see that value as explained in the first example.
   * Also, Since Strings are very popular as the HashMap key, it's important for them to be immutable so that they can retrieve the value object which was stored in HashMap.
   * Since HashMap works in the principle of hashing, which requires the same has value to function properly.
   * Mutable String would produce two different hashcodes at the time of insertion and retrieval if contents of String was modified after insertion, potentially losing the value object in the map.
3. What is String constant pool?
   * A string constant pool is a separate place in the heap memory where the values of all the strings which are defined in the program are stored.
   * When we declare a string, an object of type String is created in the stack, while an instance with the value of the string is created in the heap.
   * On standard assignment of a value to a string variable, the variable is allocated stack, while the value is stored in the heap in the string constant pool.
4. What is an exception in Java?
   * An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.



1. How does exception handling work in Java?
   * Using try-catch block or using throws keyword with method signature
2. What are exception handling keywords in Java?
   * Try {} : try block is a block of a code where there are chances of exception
   * Catch : exception thrown from try block code can be handled using catch(Exception ) method
   * There can be single or multiple catch methods
   * We can also throw new exception in catch block for method chaining or can handle the exception in catch block
   * If we want to handle separately, we can use throws keyword with method signature, which state method may throw the exception specified and whoever is going to use this method should handle this exception
3. What is the purpose of the throw and throws keywords?
   * Throw is used for throwing new exception
   * Whereas, throws is used with method signature to specify that method can throw the exception of specified kind and should be handled by method which is going to consume the current method
4. How can you handle an exception?
   * Can be handled using try catch block or redirected using throws keyword
5. Explain the Java exception hierarchy.



1. How can you catch multiple exceptions?
   * We can catch multiple exception using Specific exception type as argument to catch(Ex) method
2. What is the difference between checked and unchecked exceptions in Java?
   * All exceptions under Exception class except RuntimeException and its child are checked exceptions, also known as compile time exception
   * RuntimeException and its childs are part of unchecked exceptions, also called as run time exception
   * Checked exceptions can be propagated directly we have to use throws keyword or handle the exception, on the other hand unchecked exceptions can be automatically propagate
   * E.g. of checked Exceptions - IOException,SQLException etc. which should be handle at compile time
   * E.g. of unchecked exceptions – ArithmeticException,NullPointerException etc which can arise at run time
3. What is the difference between throw and throws keyword in Java?

|  |  |
| --- | --- |
| Throw | throws |
| throw keyword is used to throw an exception explicitly. | throws keyword is used to declare one or more exceptions, separated by commas. |
| Only single exception is thrown by using throw. | Multiple exceptions can be thrown by using throws. |
| throw keyword is used within the method. | throws keyword is used with the method signature. |
| Syntax wise throw keyword is followed by the instance variable. | Syntax wise throws keyword is followed by exception class names. |
| Checked exception cannot be propagated using throw only.Unchecked exception can be propagated using throw. | For the propagation checked exception must use throws keyword followed by specific exception class name. |

1. What is the difference between an exception and error?

| **Sr. No.** | **Key** | **Error** | **Exception** |
| --- | --- | --- | --- |
| 1 | Type | Classified as an unchecked type | Classified as checked and unchecked |
| 2 | Package | It belongs to java.lang.error | It belongs to java.lang.Exception |
| 3 | Recoverable/ Irrecoverable | It is irrecoverable | It is recoverable |
| 4 |  | It can't be occur at compile time | It can occur at run time compile time both |
| 5 | Example | OutOfMemoryError ,IOError | NullPointerException , SqlException |
| 6 |  | Developer need not handle error | Developer should handle exceptions |

1. What is the OutOfMemoryError in Java?
   * A java.lang.OutOfMemoryError is a runtime error in Java which occurs when the Java Virtual Machine (JVM) is unable to allocate an object due to insufficient space in the Java heap.
   * The Java Garbage Collector (GC) cannot free up the space required for a new object, which causes a java.lang.OutOfMemoryError.
   * This error can also be thrown when the native memory is insufficient to support the loading of a Java class.
2. What are chained exceptions in Java?
   * Chained Exception helps to identify a situation in which one exception causes another Exception in an application
   * We need to chain the exceptions to make logs readable.
3. How do you write a custom exception in Java?
   * Java custom exceptions are used to customize the exception according to user need.
   * We can create custom exception in java by extending Exception class
4. What is the difference between final, finally, and finalize in Java?

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **final** | **Finally** | **finalize** |
| 1) | Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. | Finally is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| 2) | Final is a keyword. | Finally is a block. | Finalize is a method. |

1. What happens when an exception is thrown by the main method?
   * When exception is thrown by main() method, Java Runtime **terminates** the program and print the exception message and stack trace in system console.
2. What is a try-with-resources statement?
   * The try-with-resources statement is a try statement that declares one or more resources.
   * A resource is an object that must be closed after the program is finished with it.
   * The try-with-resources statement ensures that each resource is closed at the end of the statement.
   * Any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable, can be used as a resource.

E.g. static String readFirstLineFromFile(String path) throws IOException {

**try (BufferedReader br =**

**new BufferedReader(new FileReader(path)))** {

return br.readLine();

}

}

( before try-catch block we have to close the resource manually in catch or in finally block )

1. What is a stacktrace and how does it relate to an exception?
   * a stack trace is a representation of a call stack at a certain point in time, with each element representing a method invocation. The stack trace contains all invocations from the start of a thread until the point it’s generated. This is usually a position at which an exception takes place
2. What are the advantages of Java exceptions?
   * [Advantage 1: Separating Error Handling Code from "Regular" Code](http://journals.ecs.soton.ac.uk/java/tutorial/java/exceptions/definition.html#advantageOne)
   * [Advantage 2: Propagating Errors Up the Call Stack](http://journals.ecs.soton.ac.uk/java/tutorial/java/exceptions/definition.html#advantageTwo)
   * ­­[Advantage 3: Grouping Error Types and Error Differentiation](http://journals.ecs.soton.ac.uk/java/tutorial/java/exceptions/definition.html#advantageThree)
3. Can you throw any exception inside a lambda expression’s body?
   * We can only throw exception in lambda expression if method signature present in interface is throwing that particular exception or its subtype
4. What are the rules that we need to follow when overriding a method that throws an exception?
   * When the **parent class method**doesn’t throw any exceptions, the **child class method** can’t throw any **checked exception**, but it may throw any **unchecked exceptions**.
   * When the**parent class method** throws one or more **checked exceptions**, the **child class method** can throw any **unchecked exception**.
   * When the **parent class method** has a throws clause with an **unchecked exception**, the **child class method**can throw none or any number of **unchecked exceptions**, even though they are not related.
5. What are some of the exception handling best practices?

* **Clean Up Resources in a Finally Block or Use a Try-With-Resource Statement**
* Prefer Specific Exceptions
* Throw Exceptions With Descriptive Messages
* Catch the Most Specific Exception First
* Don’t Catch Throwable
* Don’t Ignore Exceptions
* Wrap the Exception Without Consuming it
* Only add code in try block which can throw exception