Core Java

**Why is Java a platform independent language?**

[**Java language**](https://www.interviewbit.com/blog/features-of-java/) was developed in such a way that it does not depend on any hardware or software due to the fact that the [**compiler**](https://www.interviewbit.com/online-java-compiler/) compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

* The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it.

### Why is Java not a pure object oriented language?

Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure [**object oriented language**](https://www.interviewbit.com/oops-interview-questions/).

**Difference between Heap and Stack Memory in java. And how java utilizes this.**

Stack memory is the portion of memory that was assigned to every individual program. And it was fixed. On the other hand, Heap memory is the portion that was not allocated to the java program but it will be available for use by the java program when it is required, mostly during the runtime of the program.

**Java Utilizes this memory as -**

* When we write a java program then all the variables, methods, etc are stored in the stack memory.
* And when we create any object in the java program then that object was created in the heap memory. And it was referenced from the stack memory.

### Can java be said to be the complete object-oriented programming language?

It is not wrong if we claim that java is the complete object-oriented programming language. Because Everything in Java is under the classes. And we can access that by creating the objects.

But also if we say that java is not a completely object-oriented programming language because it has the support of primitive data types like int, float, char, boolean, double, etc.

Now for the question: **Is java a completely object-oriented programming language?** We can say that - Java is not a pure object-oriented programming language, because it has direct access to primitive data types. And these primitive data types don't directly belong to the Integer classes

**Difference between static methods, static variables, and static classes in java.**

* **Static Methods and Static variables** are those methods and variables that belong to the class of the java program, not to the object of the class. This gets memory where the class is loaded. And these can directly be called with the help of class names.
  + For example - We have used mathematical functions in the java program like - max(), min(), sqrt(), pow(), etc. And if we notice that, then we will find that we call it directly with the class name. Like - Math.max(), Math.min(), etc. So that is a static method.  And Similarly static variables we have used like (length) for the array to get the length. So that is the static method.
* **Static classes**- A class in the java program cannot be static except if it is the inner class. If it is an inner static class, then it exactly works like other static members of the class.

### Why is the main method static in Java?

The main method is always static because static members are those methods that belong to the classes, not to an individual object. So if the main method will not be static then for every object, It is available. And that is not acceptable by JVM. JVM calls the main method based on the class name itself. Not by creating the object.

Because there must be only 1 main method in the java program as the execution starts from the main method. So for this reason the main method is static.

**Can the static methods be overridden?**

* No! Declaration of static methods having the same signature can be done in the subclass but run time polymorphism can not take place in such cases.
* Overriding or dynamic polymorphism occurs during the runtime, but the static methods are loaded and looked up at the compile time statically. Hence, these methods cant be overridden.

**What is the main objective of garbage collection?**

The main objective of this process is to free up the memory space occupied by the unnecessary and unreachable objects during the Java program execution by deleting those unreachable objects.

* This ensures that the memory resource is used efficiently, but it provides no guarantee that there would be sufficient memory for the program execution

What is polymorphism?

Ans>>Real life example of polymorphism: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism. Polymorphism is considered one of the important features of Object-Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms. In Java polymorphism is mainly divided into two types: • Compile time Polymorphism • Runtime Polymorphism 1.Compile-time polymorphism: It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading. But Java doesn’t support the Operator Overloading. 1.Method Overloading: When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments. 2. Runtime polymorphism: It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding. Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

What is string constant pool?

String pool is nothing but a storage area in Java heap where string literals stores. It is also known as String Intern Pool or String Constant Pool. It is just like object allocation. By default, it is empty and privately maintained by the Java String class. Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also.

What is public static void main ?

main() in Java is the entry point for any Java program. It is always written as public static void main(String[] args).

public: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.

static: It is a keyword in java which identifies it is class-based.

main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as main() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.

void: It is the return type of the method. Void defines the method which will not return any value. main: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.

String args[]: It is the parameter passed to the main method.

### Difference betn Final Finally Finalize

### Final –

### 1)final is the keyword and access modifier which is used to apply restrictions on a class, method or variable. 2) Final keyword is used with the classes, methods and variables. 3) Once declared, final variable becomes constant and cannot be modified. final method cannot be overridden by sub class. final class cannot be inherited 4) Final method is executed only when we call it.

### Finally

### finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.

### Finally block is always related to the try and catch block in exception handling.

### finally block runs the important code even if exception occurs or not. finally block cleans up all the resources used in try block 4) Finally block is executed as soon as the try-catch block is executed. It's execution is not dependant on the exception.

### Finalize

### 1) finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.

### 2) finalize() method is used with the objects.

### 3) finalize method performs the cleaning activities with respect to the object before its destruction.

### 4) finalize method is executed just before the object is destroyed.

### What is JIT compiler?

**Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

What is the platform?

A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.

What are the main differences between the Java platform and other platforms?

There are the following differences between the Java platform and other platforms.

* Java is the software-based platform whereas other platforms may be the hardware platforms or software-based platforms.
* Java is executed on the top of other hardware platforms whereas other platforms can only have the hardware components.

What gives Java its 'write once and run anywhere' nature?

The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

What is classloader?

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA\_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

### What are the various access specifiers in Java?

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

* **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
* **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
* **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
* **Private** The private class, methods, or variables defined as private can be accessed within the class only.

What are the advantages of Packages in Java?

There are various advantages of defining packages in Java.

* Packages avoid the name clashes.
* The Package provides easier access control.
* We can also have the hidden classes that are not visible outside and used by the package.
* It is easier to locate the related classes.

### What is the purpose of a default constructor?

The purpose of the default constructor is to assign the default value to the objects. The java compiler creates a default constructor implicitly if there is no constructor in the class.

What do you understand by copy constructor in Java?

There is no copy constructor in java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

* By constructor
* By assigning the values of one object into another
* By clone() method of Object class

### What is ****this**** keyword in java?

The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.

What are the main uses of this keyword?

There are the following uses of **this** keyword.

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

What is the Inheritance?

Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class. It is used for Code Reusability and Method Overriding. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

There are five types of inheritance in Java.

* Single-level inheritance
* Multi-level inheritance
* Multiple Inheritance
* Hierarchical Inheritance
* Hybrid Inheritance

Multiple inheritance is not supported in Java through class.

Why is Inheritance used in Java?

There are various advantages of using inheritance in Java that is given below.

* Inheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide the specific implementation of the method.
* Runtime polymorphism cannot be achieved without using inheritance.
* We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic.
* Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.
* Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.

### Why is multiple inheritance not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.

### What is aggregation?

Aggregation can be defined as the relationship between two classes where the aggregate class contains a reference to the class it owns. Aggregation is best described as a **has-a** relationship. For example, The aggregate class Employee having various fields such as age, name, and salary also contains an object of Address class having various fields such as Address-Line 1, City, State, and pin-code.

### What is composition?

Holding the reference of a class within some other class is known as composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. In other words, we can say that composition is the particular case of aggregation which represents a stronger relationship between two objects. Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.

### What is the difference between aggregation and composition?

Aggregation represents the weak relationship whereas composition represents the strong relationship. For example, the bike has an indicator (aggregation), but the bike has an engine (composition).

### Why does Java not support pointers?

The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe(unsecured) and complex to understand.

### What is super in java?

The **super** keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.

 What are the main uses of the super keyword?

There are the following uses of super keyword.

* super can be used to refer to the immediate parent class instance variable.
* super can be used to invoke the immediate parent class method.
* super() can be used to invoke immediate parent class constructor.

What are the differences between this and super keyword?

There are the following differences between this and super keyword.

* The super keyword always points to the parent class contexts whereas this keyword always points to the current class context.
* The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor.
* The super and this must be the first statement inside constructor otherwise the compiler will throw an error.

### What is object cloning?

The object cloning is used to create the exact copy of an object. The clone() method of the Object class is used to clone an object. The **java.lang.Cloneable** interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

What is method overloading?

Method overloading is the polymorphism technique which allows us to create multiple methods with the same name but different signature. We can achieve method overloading in two ways.

* By Changing the number of arguments
* By Changing the data type of arguments

Method overloading increases the readability of the program. Method overloading is performed to figure out the program quickly.

What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

**Rules for Method overriding**

* The method must have the same name as in the parent class.
* The method must have the same signature as in the parent class.
* Two classes must have an IS-A relationship between them.

Can we change the scope of the overridden method in the subclass?

Yes, we can change the scope of the overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.

* The private can be changed to protected, public, or default.
* The protected can be changed to public or default.
* The default can be changed to public.
* The public will always remain public.

Can we modify the throws clause of the superclass method while overriding it in the subclass?

Yes, we can modify the throws clause of the superclass method while overriding it in the subclass. However, there are some rules which are to be followed while overriding in case of exception handling.

* If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception, but it can declare the unchecked exception.
* If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

### What is covariant return type?

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. The covariant return type specifies that the return type may vary in the same direction as the subclass.

### Can we declare a constructor as final?

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you try to do so, The compiler will throw an error.

### Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

 What are the advantages of Encapsulation in Java?

There are the following advantages of Encapsulation in Java?

* By providing only the setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.
* It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
* It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.
* The encapsulate class is easy to test. So, it is better for unit testing.
* The standard IDE's are providing the facility to generate the getters and setters. So, it is easy and fast to create an encapsulated class in Java.

### What is Exception Handling?

Exception Handling is a mechanism that is used to handle runtime errors. It is used primarily to handle checked exceptions. Exception handling maintains the normal flow of the program. There are mainly two types of exceptions: checked and unchecked. Here, the error is considered as the unchecked exception.

### What is the difference between Checked Exception and Unchecked Exception?

### 1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions, e.g., IOException, SQLException, etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions, e.g., ArithmeticException, NullPointerException, etc. Unchecked exceptions are not checked at compile-time.

### Why are the objects immutable in java?

Because Java uses the concept of the string literal. Suppose there are five reference variables, all refer to one object "sachin". If one reference variable changes the value of the object, it will be affected by all the reference variables. That is why string objects are immutable in java.

### What is the purpose of toString() method in Java?

The toString() method returns the string representation of an object. If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object, etc. depending upon your implementation. By overriding the toString() method of the Object class, we can return the values of the object, so we don't need to write much code.

What are the advantages of Java inner classes?

There are two types of advantages of Java inner classes.

* Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of the outer class including private.
* Nested classes are used to develop a more readable and maintainable code because it logically groups classes and interfaces in one place only.
* **Code Optimization:** It requires less code to write.

### What is a nested class?

The nested class can be defined as the class which is defined inside another class or interface. We use the nested class to logically group classes and interfaces in one place so that it can be more readable and maintainable. A nested class can access all the data members of the outer class including private data members and methods.

What are the disadvantages of using inner classes?

There are the following main disadvantages of using inner classes.

* Inner classes increase the total number of classes used by the developer and therefore increases the workload of JVM since it has to perform some routine operations for those extra classes which result in slower performance.
* IDEs provide less support to the inner classes as compare to the top level classes and therefore it annoys the developers while working with inner classes.

### What are anonymous inner classes?

Anonymous inner classes are the classes that are automatically declared and instantiated within an expression. We cannot apply different access modifiers to them. Anonymous class cannot be static, and cannot define any static fields, method, or class. In other words, we can say that it a class without the name and can have only one object that is created by its definition.

### How is garbage collection controlled?

Garbage collection is managed by JVM. It is performed when there is not enough space in the memory and memory is running low. We can externally call the System.gc() for the garbage collection. However, it depends upon the JVM whether to perform it or not.

### What is serialization?

Serialization in Java is a mechanism of writing the state of an object into a byte stream. It is used primarily in Hibernate, RMI, JPA, EJB and JMS technologies. It is mainly used to travel object's state on the network (which is known as marshaling). Serializable interface is used to perform serialization. It is helpful when you require to save the state of a program to storage such as the file. At a later point of time, the content of this file can be restored using deserialization. It is also required to implement RMI(Remote Method Invocation). With the help of RMI, it is possible to invoke the method of a Java object on one machine to another machine.

### What is Deserialization?

Deserialization is the process of reconstructing the object from the serialized state. It is the reverse operation of serialization. An ObjectInputStream deserializes objects and primitive data written using an ObjectOutputStream

### What is the transient keyword?

If you define any data member as transient, it will not be serialized. By determining transient keyword, the value of variable need not persist when it is restored.

### What is Externalizable?

The Externalizable interface is used to write the state of an object into a byte stream in a compressed format. It is not a marker interface.

### What are wrapper classes?

Wrapper classes are classes that allow primitive types to be accessed as objects. In other words, we can say that wrapper classes are built-in java classes which allow the conversion of objects to primitives and primitives to objects. The process of converting primitives to objects is called autoboxing, and the process of converting objects to primitives is called unboxing. There are eight wrapper classes present in **java.lang** package .

What are autoboxing and unboxing? When does it occur?

The autoboxing is the process of converting primitive data type to the corresponding wrapper class object, eg., int to Integer. The unboxing is the process of converting wrapper class object to primitive data type. For eg., integer to int. Unboxing and autoboxing occur automatically in Java. However, we can externally convert one into another by using the methods like valueOf() or xxxValue().

It can occur whenever a wrapper class object is expected, and primitive data type is provided or vice versa.

* Adding primitive types into Collection like ArrayList in Java.
* Creating an instance of parameterized classes ,e.g., ThreadLocal which expect Type.
* Java automatically converts primitive to object whenever one is required and another is provided in the method calling.
* When a primitive type is assigned to an object type.

### What is object cloning?

The object cloning is a way to create an exact copy of an object. The clone() method of the Object class is used to clone an object. The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException. The clone() method is defined in the Object class.

What are the advantages and disadvantages of object cloning?

**Advantage of Object Cloning**

* You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method.
* It is the easiest and most efficient way of copying objects, especially if we are applying it to an already developed or an old project. Just define a parent class, implement Cloneable in it, provide the definition of the clone() method and the task will be done.
* Clone() is the fastest way to copy the array.

**Disadvantage of Object Cloning**

* To use the Object.clone() method, we have to change many syntaxes to our code, like implementing a Cloneable interface, defining the clone() method and handling CloneNotSupportedException, and finally, calling Object.clone(), etc.
* We have to implement the Cloneable interface while it does not have any methods in it. We have to use it to tell the JVM that we can perform a clone() on our object.
* Object.clone() is protected, so we have to provide our own clone() and indirectly call Object.clone() from it.
* Object.clone() does not invoke any constructor, so we do not have any control over object construction.
* If you want to write a clone method in a child class, then all of its superclasses should define the clone() method in them or inherit it from another parent class. Otherwise, the super.clone() chain will fail.
* Object.clone() supports only shallow copying, but we will need to override it if we need deep cloning.

Collection Framework

 What is the Collection framework in Java?

Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.

### What are the main differences between array and collection?

Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data, but they differ in many ways. The main differences between the array and Collection are defined below:

* Arrays are always of fixed size, i.e., a user can not increase or decrease the length of the array according to their requirement or at runtime, but In Collection, size can be changed dynamically as per need.
* Arrays can only store homogeneous or similar type objects, but in Collection, heterogeneous objects can be stored.
* Arrays cannot provide the ?ready-made? methods for user requirements as sorting, searching, etc. but Collection includes readymade methods to use.

### What is the difference between ArrayList and Vector?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. |
| 4) | ArrayList is not ?thread-safe? as it is not synchronized. | Vector list is ?thread-safe? as it?s every method is synchronized. |

### What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses a doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because too much is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |
| 4) | ArrayList provides random access. | LinkedList does not provide random access. |
| 5) | ArrayList takes less memory overhead as it stores only object | LinkedList takes more memory overhead, as it stores the object as well as the address of that object. |

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | The Iterator traverses the elements in the forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | The Iterator can be used in List, Set, and Queue. | ListIterator can be used in List only. |
| 3) | The Iterator can only perform remove operation while traversing the collection. | ListIterator can perform ?add,? ?remove,? and ?set? operation while traversing the collection. |

What is the difference between List and Set?

The List and Set both extend the collection interface. However, there are some differences between the both which are listed below.

* The List can contain duplicate elements whereas Set includes unique items.
* The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order.
* The List interface contains a single legacy class which is Vector class whereas Set interface does not have any legacy class.
* The List interface can allow n number of null values whereas Set interface only allows a single null value.

### What are the advantages of the Collection framework?

Following are the advantages of the Collection framework:-

**Consistent API**: The API has a core set of interfaces like Collection, Set, List, or Map, and all the classes (ArrayList, LinkedList, Vector, and so on) that implement these interfaces have some common set of methods.

**Cuts programming effort**: Instead of worrying about the Collection's design, a programmer may concentrate on how best to use it in his program. As a result, the fundamental principle of Object-oriented programming (i.e. abstraction) has been applied successfully.

**Improves program speed and quality** by offering high-performance implementations of useful data structures and algorithms, as the programmer does not have to worry about the optimum implementation of a certain data structure in this scenario. They can simply use the best implementation to improve the performance of their program significantly.

### Explain the various interfaces used in the Collection framework.

The collection framework has several interfaces, each of which is used to store a different sort of data. The interfaces included in the framework are listed below.

**1. Iterable Interface**: This is the collection framework's primary interface. The iterable interface is extended by the collection interface. As a result, all interfaces and classes implement this interface by default. This interface's main purpose is to provide an iterator for the collections. As a result, this interface only has one abstract method, the iterator.

**2. Collection Interface**: The collection framework's classes implement this interface, which extends the iterable interface. This interface covers all of the basic methods that every collection has, such as adding data to the collection, removing data from the collection, clearing data, and so on. All of these methods are incorporated in this interface because they are used by all classes, regardless of their implementation style. Furthermore, including these methods in this interface guarantees that the method names are consistent across all collections. In summary, we may conclude that this interface lays the groundwork for the implementation of collection classes.

**3. List Interface**: The collection interface has a child interface called the list interface. This interface is devoted to list data, in which we can store all of the objects in an ordered collection. This also allows for the presence of redundant data. Various classes, such as ArrayList, Vector, Stack, and others, implement this list interface. We can create a list object with any of these classes because they all implement the list.

**4. Queue Interface**: A queue interface, as the name implies, follows the FIFO (First In First Out) order of a real-world queue line. This interface is for storing all elements in which the order of the elements is important. When we try to shop at a store, for example, the bills are issued on a first-come, first-served basis. As a result, the individual whose request is first in line receives the bill first. PriorityQueue, Deque, ArrayDeque, and other classes are available. Because all of these subclasses implement the queue, we can use any of them to create a queue object.

**5. Deque Interface**: It differs slightly from the queue data structure.  Deque, also known as a double-ended queue, is a data structure in which elements can be added and removed from both ends. The queue interface is extended by this interface. ArrayDeque is the class that implements this interface. Because this class implements the deque, we can use it to create a deque object.

**6. Set Interface**: A set is an unordered group of objects in which duplicate values cannot be kept. This collection is utilised when we want to avoid duplication of things and only keep the ones that are unique. Various classes, such as HashSet, TreeSet, LinkedHashSet, and others, implement this set interface. We can create a set object with any of these classes because they all implement the set.

**7. Sorted Set Interface**: This interface resembles the set interface in appearance. The only difference is that this interface provides additional methods for maintaining element ordering. The sorted set interface is an extension of the set interface that is used to manage sorted data. TreeSet is the class that implements this interface. We can create a SortedSet object using this class because it implements the SortedSet interface.

### Differentiate between HashSet and HashMap.

**HashSet** is a Set Interface implementation that does not allow duplicate values. The essential point is that objects stored in HashSet must override equals() and hashCode() methods to ensure that no duplicate values are stored in our set.

**HashMap** is a Map Interface implementation that maps a key to a value. In a map, duplicate keys are not permitted.

| **HashSet** | **HashMap** |
| --- | --- |
| It implements the Set Interface. | It implements the Map Interface. |
| It does not allow duplicate values. | The key needs to be unique while two different keys can have the same value. |
| While adding an element it requires only one object as a parameter. | While adding an entry, it requires two object values, the **Key**and the **Value**as the parameter. |
| Internally, HashSet uses HashMap to add entries. The key K in a HashSet is the argument supplied in the add(Object) method. For each value supplied in the add(Object) method, Java assigns a dummy value. | There is no concept of duplicate values. |
| It is slower than HashMap. | It is faster than HashSet. |
| It uses the add() method for adding elements. | It uses the put() method for adding data elements. |

### What is the default size of the load factor in hashing based collection?

The default load factor size is **0.75**. The default capacity is calculated by multiplying the initial capacity by the load factor

 What is the difference between Collection and Collections?

The differences between the Collection and Collections are given below.

* The Collection is an interface whereas Collections is a class.
* The Collection interface provides the standard functionality of data structure to List, Set, and Queue. However, Collections class is to sort and synchronize the collection elements.
* The Collection interface provides the methods that can be used for data structure whereas Collections class provides the static methods which can be used for various operation on a collection.

### Why we override equals() method?

The equals method is used to check whether two objects are the same or not. It needs to be overridden if we want to check the objects based on the property.

For example, Employee is a class that has 3 data members: id, name, and salary. However, we want to check the equality of employee object by the salary. Then, we need to override the equals() method.

What is the advantage of the generic collection?

There are three main advantages of using the generic collection.

* If we use the generic class, we don't need typecasting.
* It is type-safe and checked at compile time.
* Generic confirms the stability of the code by making it bug detectable at compile time.

[How to remove duplicates from ArrayList?](https://www.javatpoint.com/how-to-remove-duplicates-from-arraylist-in-java)

There are two ways to remove duplicates from the ArrayList.

* **Using HashSet:** By using HashSet we can remove the duplicate element from the ArrayList, but it will not then preserve the insertion order.
* **Using LinkedHashSet:** We can also maintain the insertion order by using LinkedHashSet instead of HashSet.

The Process to remove duplicate elements from ArrayList using the LinkedHashSet:

* Copy all the elements of ArrayList to LinkedHashSet.
* Empty the ArrayList using clear() method, which will remove all the elements from the list.
* Now copy all the elements of LinkedHashset to ArrayList.