**java.lang.Object** is the super class in java for all the classes.

In Java, the flow of a execution is called **Thread**

Java uses **Just In Time** compiler to enable high performance. JIT is used to convert the instructions into bytecodes.

The **constructor** is a method which has the same name as class name. Constructor can be overloaded.

All Java codes are defined in a class. A Class has variables and methods.

**Variables** are attributes which define the state of a class.

An instance of a class is called object. The object has state and behavior.

**Class** can't be private and protected.

**Methods** are the place where the exact business logic has to be done.

It contains a set of statements (or) instructions to satisfy the particular requirement.

**Access modifiers**

**Default** - when nothing is mentioned it is default Its limited to the package only.

**Private** - methods and data members are only accessible within the class.Class and interface can't be declared as private.

If a class has private constructor then you cannot create the object of that class from outside of the class.

**Protected** - Protected data member and method are only accessible by the classes of the same package and the subclasses present in any package.

classes cannot be declared as a protected.

**Public** - The members, methods and classes that are declared public can be accessed from anywhere. This modifier doesn’t put any restriction on the access.

**Non Access modifiers**(static, final, abstract, synchronized, transient, volatile, strictfp)

**Static** - When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

Static variables, static blocks, static members.(This or super keyword can't be used)

**Final** - variable(When a variable is declared with final keyword, its value can’t be modified),

**class**(When a class is declared with final keyword, it is called a final class. A final class cannot be extended(inherited).All the wrapper classes are final)

The other use of final with classes is to create an immutable class like the predefined String class.You can not make a class immutable without making it final.

**Methods** - When a method is declared with final keyword, it is called a final method. A final method cannot be overridden. The Object class does this—a number of its methods are final.

**abstract** (mobile button example)- in Java, an instance of an abstract class cannot be created, we can have references of abstract class type though.

an **abstract class** can contain constructors in Java. And a constructor of abstract class is called when an instance of a inherited class is created.

In Java, we can have an abstract class without any abstract method. This allows us to create classes that cannot be instantiated, but can only be inherited.

Abstract classes can also have final methods (methods that cannot be overridden)

Oops concept basically has four important feautures

**Inheritance** - Inheritance means one class can extend to another class. So that the codes can be reused from one class to another class.

**Encapsulation** is achieved when each object keeps its state private, inside a class.

Other objects don’t have direct access to this state.

Instead, they can only call a list of public functions — called methods.

**Polymorphism** - A single object can refer the super class or sub-class depending on the reference type which is called polymorphism.

Polymorphism is the ability of an object to take on many forms

**Method overloading** is an example of static polymorphism, while method overriding is an example of dynamic polymorphism

**Method overloading** is a feature that allows a class to have more than one method having the same

**Method Overriding** (Run time polymorphism) is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.

**abstraction** - Abstraction is a process of hiding the implementation details from the user. Оnly the functionality will be provided to the user. In Java, abstraction is achieved using abstract classes and interfaces

**Interface** - Multiple inheritance cannot be achieved in java. To overcome this problem Interface concept is introduced.

An interface is a template which has only method declarations and not the method implementation.

**Final** is a modifier applicable for classes methods and variables.

**variable final** = constant variable we can't reassign any other variables.

**method Final** = in child class we can't override the medhod in the child class.

**Class Final** = We can't extend the class(to create child class)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Finally** is a block always associated with try catch.

Try - risky code

**catch** - handling code

**final** - clean up activities.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Finalize()** is a method which is always invoked by garbage collector

just before destroying an object to perform clean up activities.

**Finally** meant for cleanup activities related to try block.

whereas finalize() meant for clean up activities related to object.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Difference between **string and string buffer**

String objects are immutable

String buffer are mutable

String object once created, we can't make any change to the object.

Stringbuffer created, value can be changed

1.once we create a string object we can't perform any changes in the existing object. if we are trying to perform any changes with those changes a new

object will be created. this non changeable nature is nothing but immutability of the string object.

2. Once we creates a StringBuffer object we can perform any type of changes in

the existing object. this changeable is ntohing but mutuability of the Stringbuffer Object.

3. StringBuilder : This is same as String Buffer except for the String Builder which is not threaded safety that is not synchronized.

So obviously performance is fast.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Difference between == and equals.**

**==** meant for reference comparison or address comparison

**.equals** meant for content comparison.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

difference between overrloading and overriding.

**overrloading** - same method with different parameters.

**overriding** - Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String buffer synchronized, Thread-safe, waiting time of threads and performance is low.

String builder non synchronized, Not thread-safe, performance is faster.

Yeah if the content is fixed and won't change frequently then we should go for string.

If the content is not fixed and keep on changing but Thread safety is required then we should go for String Buffer.

If the content is not fixed and keep on changing and thread safety is not required then we should go for string builder.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Interface** (building plan)- If we just have requirements then we should go for Interface, eg.servlet.

**abstract class** (partially completed building) - we have only partial implementation in the interface. eg. generic servlet & httpservlet.

**concrete class** -(completed building) complete implementation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Difference between exception and error.

Throwable acts as root for java exception hierarchy.

exceptions are caused by our program and it is recoverable.

error caused by the lack of system resources.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(Hall ticket scenarios)

The exceptions which are checked by the compiler is called checked exceptions, whether programmer is handling or not.FileNotfoundexception

The exceptions which are not checked by the compiler is called unchecked exceptions,Arithmeticexception

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Exception handling

exception is a runtime error which interrupts the normal flow of program.

exception handling is a set of code which handles the exception.

exceptions in java can be handled by try catch and declare the exception using throws keyword.

* The normal flow of the execution won’t be terminated if exception got handled
* We can identify the problem by using catch declaration

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

difference between throw and throws

Throw - used to throw exception|used inside method or static block initializer.|only one exception

Throws - used for declaration for exception.|used in method declaration | multiple exceptions

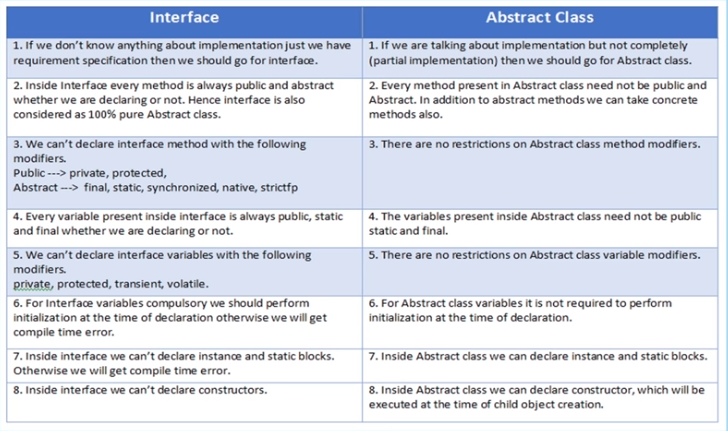
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Try - If there is no exception it will be executed.

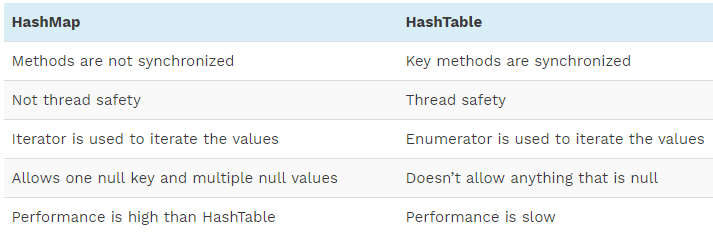
Catch - if there is exception It will be executed

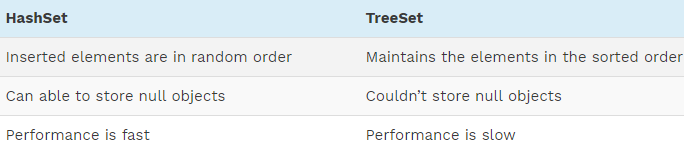
Finally - It is compulsory to be executed.The finally block always executes when the try block exits.

difference between Interface and abstract class.



Difference between hashmap and hashtable





**Q** **#24)  What is mean by Collections in Java?**

**Ans:** Collection is a framework that is designed to store the objects and manipulate the design to store the objects.

**Collections are used to perform the following operations:**

* Searching
* Sorting
* Manipulation
* Insertion
* Deletion

A group of objects is known as collections. All the classes and interfaces for collecting are available in Java utile package.

**Q #25) What are all the Classes and Interfaces that are available in the collections?**

**Ans:** **Given below are the Classes and Interfaces that are available in Collections:**

**Interfaces:**

* Collection
* List
* Set
* Map
* Sorted Set
* Sorted Map
* Queue

**Classes:**

* Lists:
* Array List
* Vector
* Linked List

**Sets:**

* Hash set
* Linked Hash Set
* Tree Set

**Maps:**

* Hash Map
* Hash Table
* Tree Map
* Linked Hashed Map

**Queue:**

* Priority Queue

**Q #26) What is meant by Ordered and Sorted in collections?**

**Ans:**

**Ordered:**

It means the values that are stored in a collection is based on the values that are added to the collection. So we can iterate the values from the collection in a specific order.

**Sorted:**

Sorting mechanism can be applied internally or externally so that the group of objects sorted in a particular collection is based on properties of the objects.

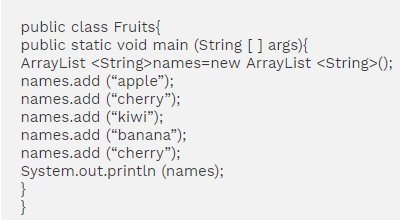
**Q #27) Explain about the different lists available in the collection.**

**Ans:**Values added to the list is based on the index position and it is ordered by index position. Duplicates are allowed.

**Types of Lists are:**

**Array List:**

* Fast iteration and fast Random Access.
* It is an ordered collection (by index) and not sorted.
* It implements Random Access Interface.

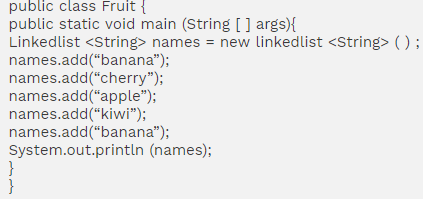


[Apple, cherry, kiwi, banana, cherry]

From the output, Array List maintains the insertion order and it accepts the duplicates. But not sorted.

**Linked List:**

* Elements are doubly linked to one another.
* Performance is slow than Array list.
* Good choice for insertion and deletion.
* In Java 5.0 it supports common queue methods peek( ), Pool ( ), Offer ( ) etc.
* Maintains the insertion order and accepts the duplicates.



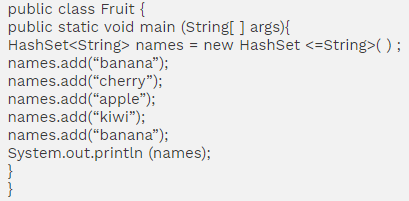
[ banana,cherry,apple,kiwi,banana]

Maintains the insertion order and accepts the duplicates.

**Set c**ares about uniqueness. It doesn’t allow duplications. Here “equals ( )” method is used to determine whether two objects are identical or not.

**Hash Set:**

* Unordered and unsorted.
* Uses the hash code of the object to insert the values.
* Use this when the requirement is “no duplicates and don’t care about the order”.

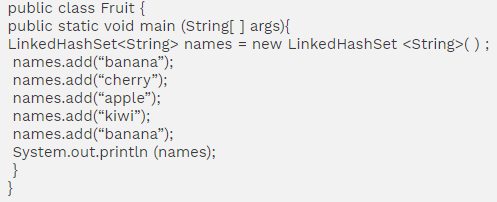


[banana, cherry, kiwi, apple]

Doesn’t follow any insertion order. Duplicates are not allowed.

**Linked Hash set:**

* An ordered version of the hash set is known as Linked Hash Set.
* Maintains a doubly-Linked list of all the elements.
* Use this when the iteration order is required.

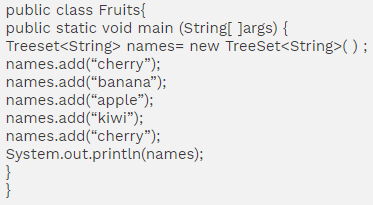


[banana, cherry, apple, kiwi]

Maintains the insertion order in which they have been added to the Set. Duplicates are not allowed.

**Tree Set:**

* It is one of the two sorted collections.
* Uses “Read-Black” tree structure and guarantees that the elements will be in an ascending order.
* We can construct a tree set with the constructor by using comparable (or) comparator.



**Output:**

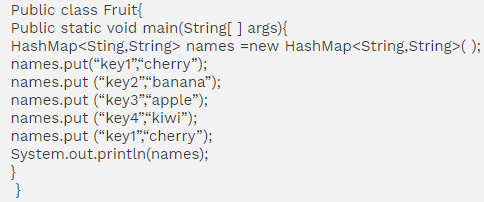
[apple, banana, cherry, kiwi]

TreeSet sorts the elements in an ascending order. And duplicates are not allowed.

**Map** cares about unique identifier. We can map a unique key to a specific value. It is a key/value pair. We can search a value, based on the key. Like set, Map also uses “equals ( )” method to determine whether two keys are same or different.

**Hash Map:**

* Unordered and unsorted map.
* Hashmap is a good choice when we don’t care about the order.
* It allows one null key and multiple null values.



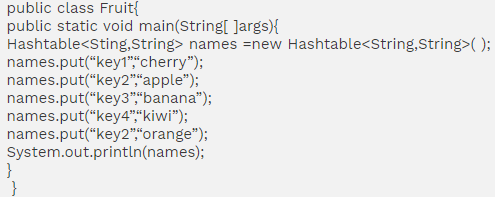
{key2 =banana, key1=cherry, key4 =kiwi, key3= apple}

Duplicate keys are not allowed in Map.

Doesn’t maintain any insertion order and is unsorted.

**Hash Table:**

* Like vector key, methods of the class are synchronized.
* Thread safety and therefore slows the performance.
* Doesn’t allow anything that is null.

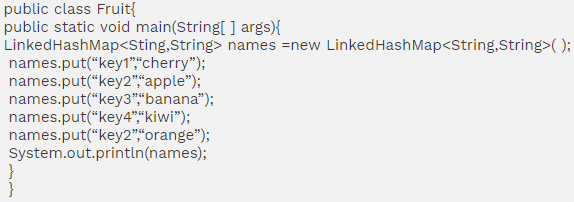


{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

**Linked Hash Map:**

* Maintains insertion order.
* Slower than Hash map.
* Can expect a faster iteration.

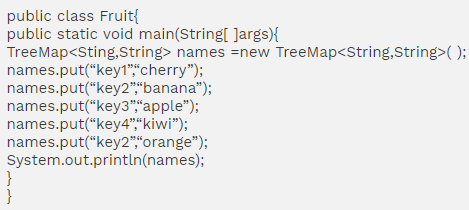


{key2=apple, key1=cherry,key4=kiwi, key3=banana}

Duplicate keys are not allowed.

**TreeMap:**

* Sorted Map.
* Like Tree set, we can construct a sort order with the constructor.



{key1=cherry, key2=banana, key3 =apple, key4=kiwi}

It is sorted in ascending order based on the key. Duplicate keys are not allowed.