|  |  |  |
| --- | --- | --- |
|  | |  |
| **Abstract Class** | | **Interface** |
| Extend only one abstract class. Support multiple inheritance. Contains Data Member, constructors. | | Implement multiple interfaces. Does not support multiple inheritance. Does not contain Data Member, constructors. |
| **String** | | **String Buffer** |
| Immutable, Thread-safe, string constant pool, synchronization. String is slow and consumes more memory when we concatenate too many strings because every time it creates new instance. | | Mutable, Thread-safe, heap memory, synchronization. String Buffer is fast and consumes less memory when concatenate too many strings. |
| **String Buffer** | **String Builder** | |
| Synchronized i.e., thread safe. It means two threads can’t call the methods of String Buffer simultaneously. | Non-synchronized i.e., not thread safe. It means two threads can call the methods of String Builder simultaneously. | |

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| Contain both primitives and objects. Not Type Safety i.e. holds same data type, homogeneous. Array does not provides remove an element. Use for loop or for each loop through iterate the Array. | Contains only objects. Type Safety means generics. Array does not provides remove an element. Use iterate through iterate ArrayList. Maintain insertion order. |
| **ArrayList** | **Linked List** |
| Implements List. ArrayList internally uses a **dynamic array** to store the elements. Manipulation with ArrayList is **slow** because it internally uses an array. ArrayList is **better for storing and accessing** data. | implements List and Deque interfaces. LinkedList internally uses a **doubly linked list** to store the elements. Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list. LinkedList is **better for manipulating** data. |
| **ArrayList** | **Vector** |
| **Not Synchronized.** Faster. ArrayList **increments 50%** of current array size if the number of elements exceeds from its capacity. Is not a legacy class introduced in JDK 1.2? | **Synchronized i.e., slow.** Vector **increments 100%** means doubles the array size if the total number of elements exceeds than its capacity. Legacy class. |
| **Hash Map** | **Hash Set** |
| Implementation of Map interface. Store data in form of key value pair. Hash Map is faster because unique key. Put method | Implementation of Set interface. Store only object. Slower. Add method is used for add element. |

**HashMap and Hashtable both are used to store data in key and value form. Both are using hashing technique to store unique keys.**

|  |  |
| --- | --- |
| **HashMap** | **Hash-Table** |
| **Non synchronized**. **Allows one null key and multiple null values**. Fast. Iterator in HashMap is **fail-fast**. Inherits **Abstract Map** class. It maintains order. When one thread is adding or removing the value and at the same time T2 T3 T4 try to access particular value that value is gone from that object….immediately it gives you concurrent modification error. | **Synchronized**. **Doesn't allow any null key or value**. Slow. Legacy Class. Enumerator in Hash table is **not fail-fast**. Inherits **Dictionary** class. |
| **List** | **ArrayList** |
| **Interface, cannot be instantiated. List interface extends collection frame work.** | **Class, can be instantiated. Extend abstract list class and implement list interface.** |
| **List** is a type of **ordered collection** that maintains the elements in insertion order while **Set** is a type of unordered collection so elements are not maintained any order. **List** allows duplicates while **Set** doesn't allow duplicate elements. | |

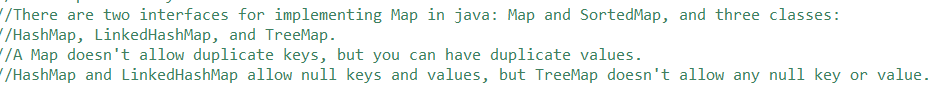












**duplicate value is not allowed. Accept null**

 When object is created, java provide a particular number for the object it is called Hash Code and it like 32 bits.

**Q. Difference between object and class?**

|  |  |
| --- | --- |
| **Object** | **Class** |
| Object is an **instance** of a class. | Class is a **blueprint or template** from which objects are created. |
| Object is a **real-world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a **group of similar objects**. |
| Object is a **physical** entity. | Class is a **logical** entity. |
| Object is created through **new keyword** mainly e.g., Student s1=new Student (); | Class is declared using **class keyword** e.g., class Student {} |
| Object is created **many times** as per requirement. | Class is declared **once**. |
| Object **allocates memory when it is created**. | Class **doesn't allocated memory when it is created**. |
| There are **many ways to create object** in java such as new keyword, new Instance () method, clone () method, factory method and deserialization. | There is only **one way to define class** in java using class keyword. |

**Q. Difference between constructor and method?**

|  |  |
| --- | --- |
| **Constructor** | **Method** |
| The constructor’s name must be same as the class name. | The method name may or may not be same as the class name. |
| A constructor must not have a return type not even void. | A method must have a return type. |
| A constructor is used to initialize the object. | A method is used to expose the behavior of an object. |