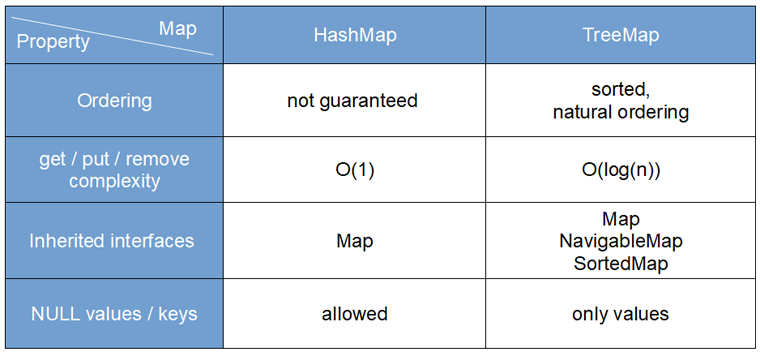
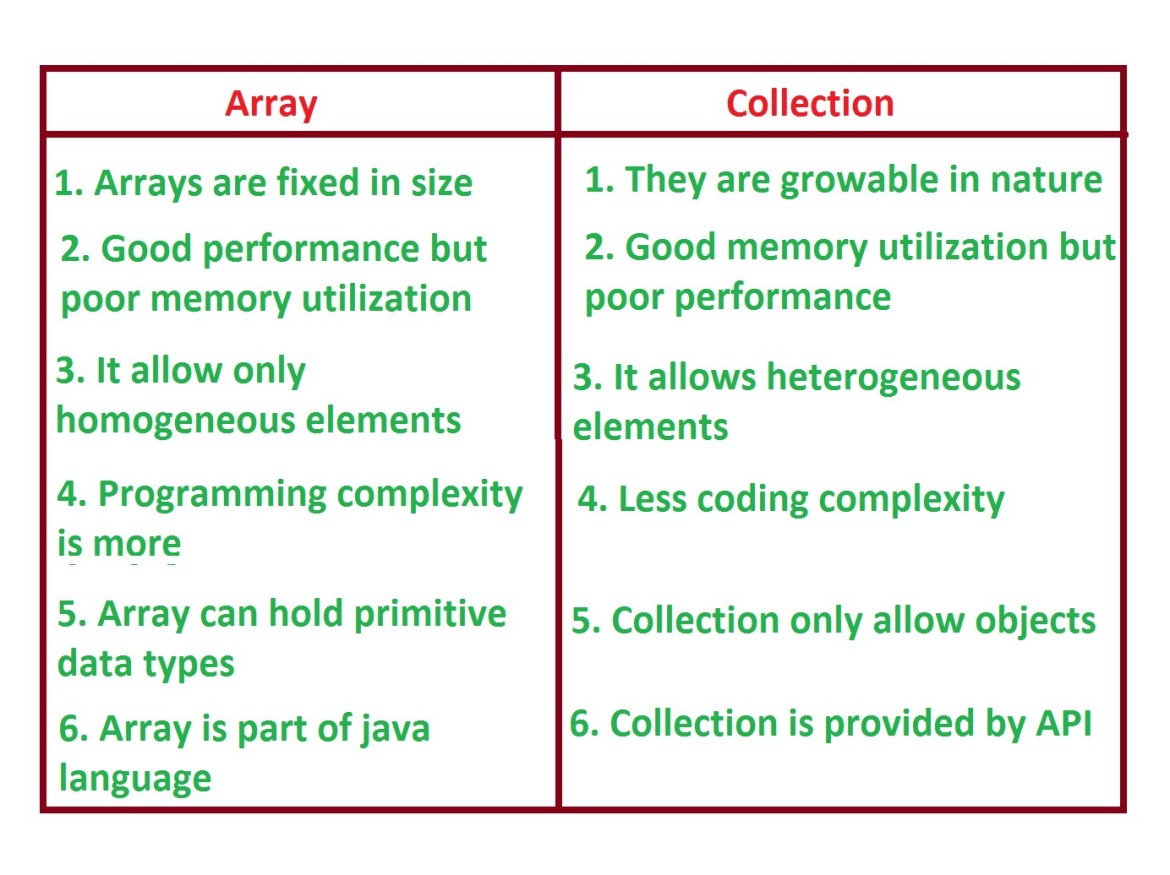
**Differences**

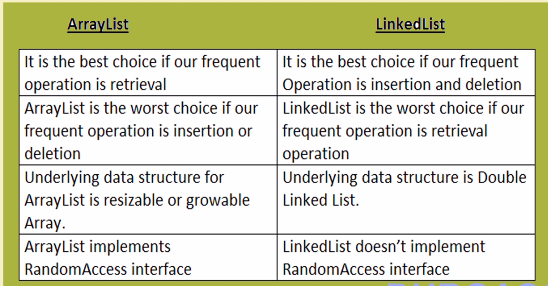
1. HashMap vs TreeMap



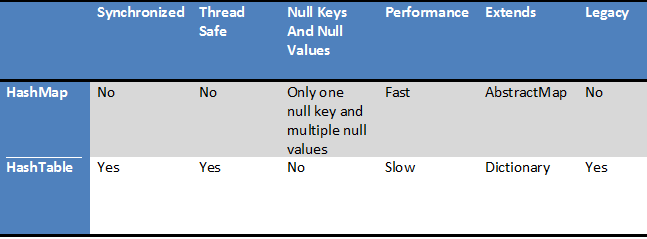
1. Array vs Collection



1. LinkedList vs ArrayList



1. HashMap vs HashTable

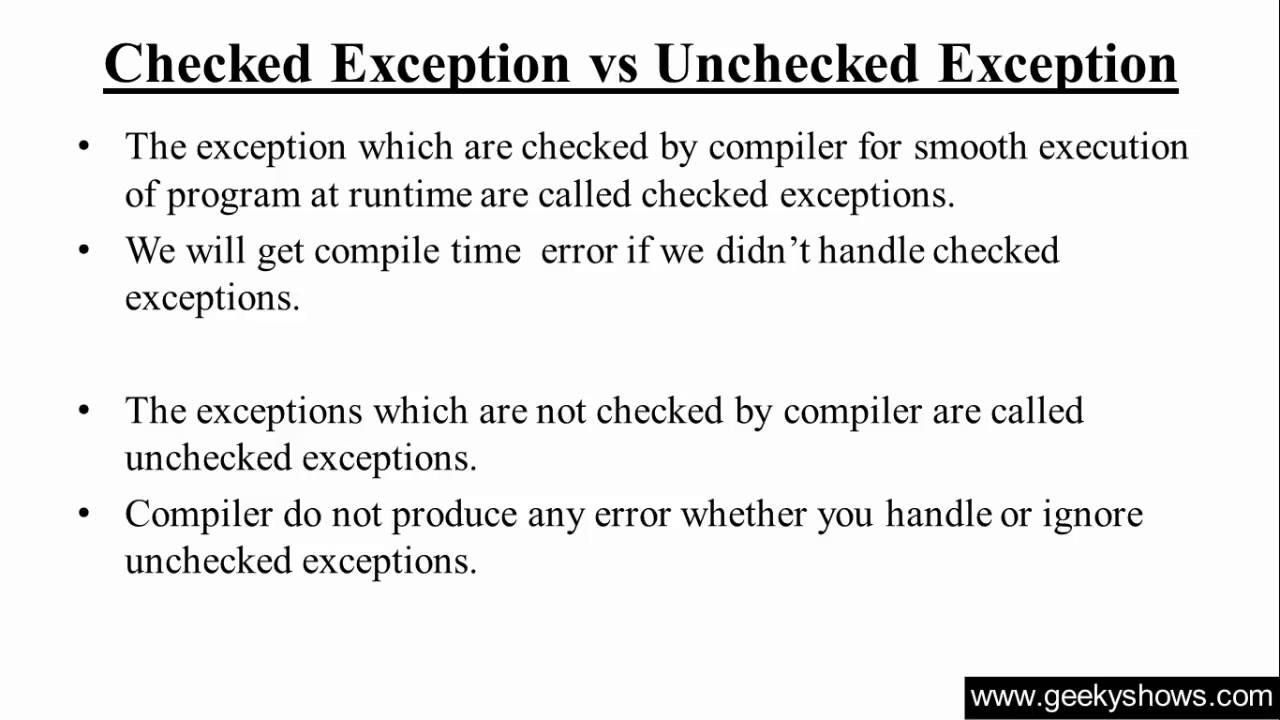


1. == vs .equals()

== => reference comparison

.equals() => content comparison

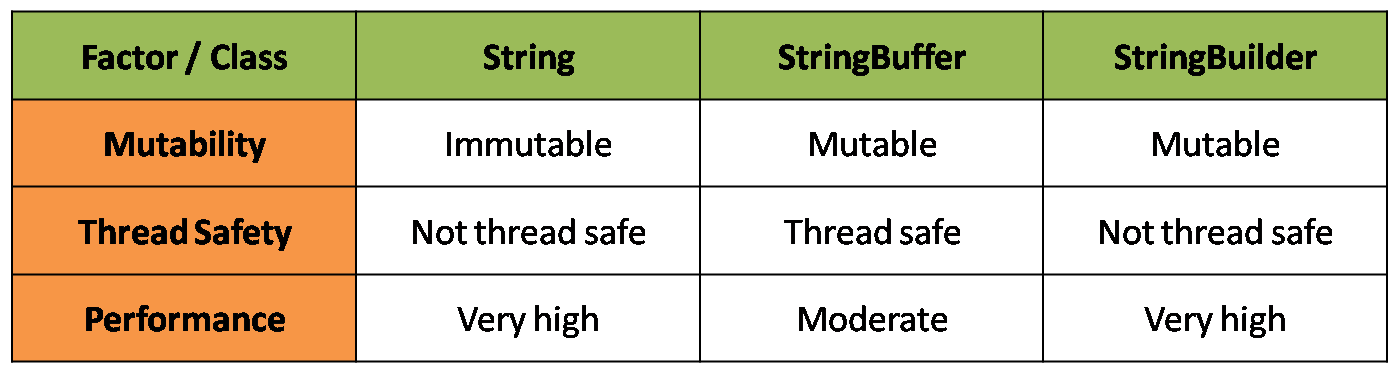
1. Checked vs Unchecked Exception



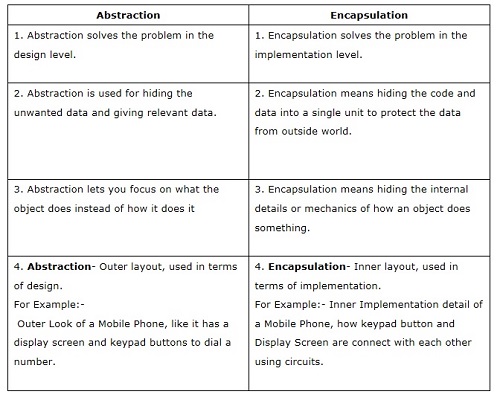
1. Thread vs Process

|  |  |
| --- | --- |
| Process | Thread |
| 1. Process means any program is in execution | 1. Part of a process |
| 1. Process takes more time | 1. Takes less time |
| 1. Process is less efficient in term of communication. | 1. More efficient |
| 1. Consumes more resources | 1. Consumes less resources |
| 1. Heavy weight | 1. Light weight |
| 1. Process is isolated | 1. Threads share memory |

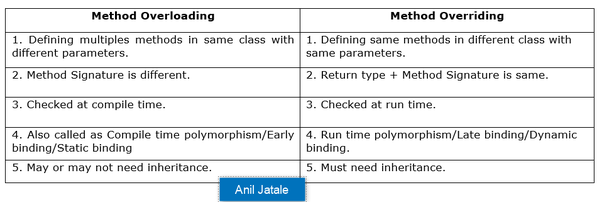
1. String vs StringBuffer vs StringBuilder



1. Abstraction vs Encapsulation



1. Overloading Vs Overriding



1. Vector vs ArrayList

Graphical user interface, text, application

Description automatically generated

1. Final vs finally vs finalize

|  |  |  |
| --- | --- | --- |
| Final | Finally | Finalize |
| Keyword | Block | Method |
| 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. |

17. Comparable vs Comparator

|  |  |
| --- | --- |
| Comparable | Comparator |
| 1. Java.lang package | 1. Java.util package |
| 1. Default natural sorting order | 1. Customized sorting order |
| 1. compareTo() | 1. compare() & equals() |

18. Runnable vs Callable

|  |  |
| --- | --- |
| Runnable | Callable |
| 1. java.lang package | 1. Java.util.concurrent package |
| 1. A runnable object doesn’t return any value | 1. May return value |
| 1. Can’t throw checked exception | 1. Can throw exception |
| 1. Introduce in 1.0 | 1. In 1.5 |
| 1. run() method | 1. call() method |

19. sleep() vs wait()

|  |  |
| --- | --- |
| Sleep() | Wait() |
| 1. Thread class | 1. Object class |
| 1. Called from anywhere | 1. Called from only synchronized block |
| 1. Doesn’t release lock | 1. Releases lock |
| 1. Awaken by interrupt() or time expires | 1. Awaken by notify() or notifyAll() |

20. new() vs newInstance()

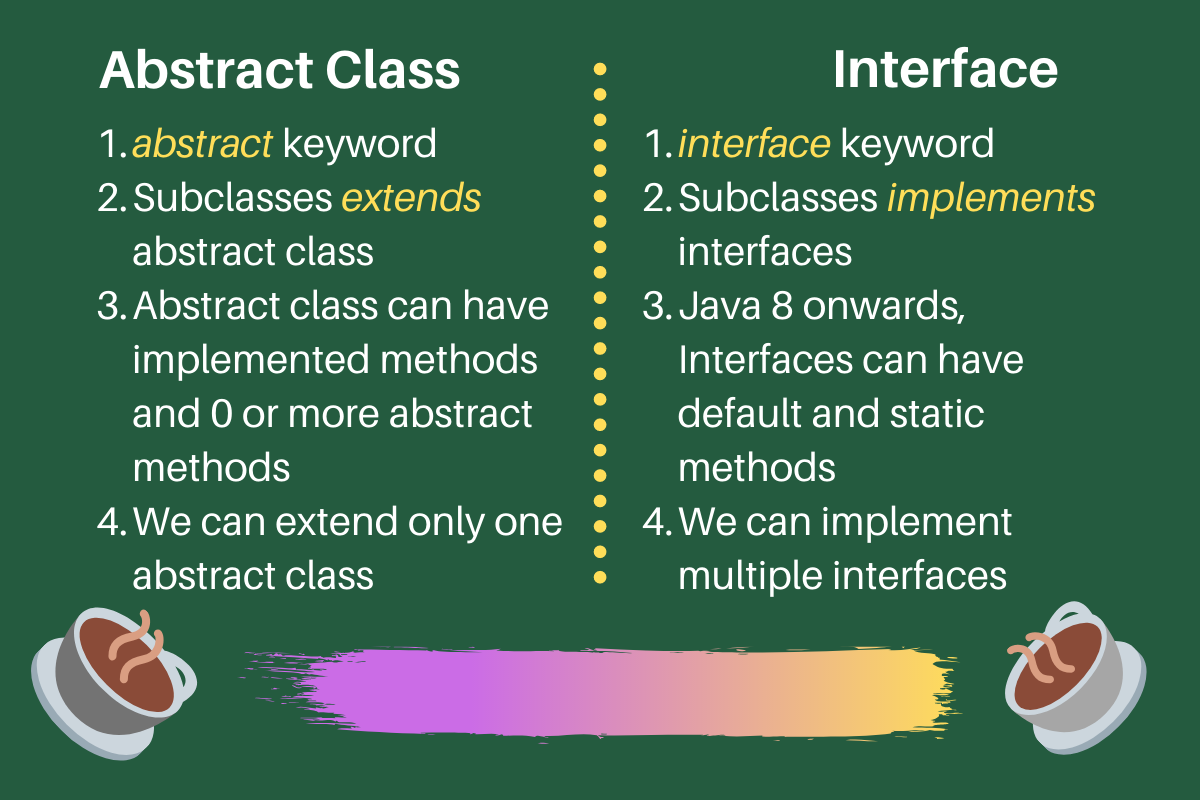
new() => object creation by using constructor

**NoClassDefFoundError** is an error that occurs when a particular class is present at compile time, but was missing at run time

newInstance() => If we want to decide type of object to be created at runtime. In this case, we have to use newInstance() method

**ClassNotFoundException** is an exception that occurs when you try to load a class at run time using **Class.forName()**

21. Abstract class vs Interface



22. Overriding vs Method Hiding

