# **Mutable and Immutable Objects**

**What is Mutable object?** The objects in which you can change the fields and states after the object is created are known as Mutable objects. ***Example***: java.util.Date, StringBuilder, and etc.

**What is Immutable object?** The [objects](https://www.edureka.co/blog/java-object/) in which you cannot change anything once the object is created are known as Immutable objects. ***Example***: Boxed primitive objects like Integer, Long and etc.

|  |  |
| --- | --- |
| **Mutable** | **Immutable** |
| Fields can be changed after the object creation | Fields cannot be changed after object creation |
| Generally provides a method to modify the field value | Does not have any method to modify the field value |
| Has Getter and Setter methods | Has only Getter method |
| Example: StringBuilder, java.util.Date | Example: String, Boxed primitive objects like Integer, Long and etc |

**How to create a Mutable class?**

To create a mutable class in Java you have to make sure the following requirements are satisfied:

1. [Provide a method](https://www.edureka.co/blog/method-overloading-and-overriding-in-java/) to modify the field values
2. Getter and Setter method

**How to create an Immutable class?**

1. A class should be declared as [final](https://www.edureka.co/blog/final-finally-and-finalize-in-java/) so that it can’t be extended.
2. All the fields should be made private so that direct access is not allowed
3. No setter methods
4. Make all mutable fields final, so that they can be assigned only once.

package com.mkyong;

// make this class final, no one can extend this class

public final class ImmutableExample {

private String name;

ImmutableExample (String name) {

this.name = name;

}

public String getName() {

return name;

}

//no setter

public static void main(String[] args) {

ImmutableExample obj = new ImmutableExample("mkyong");

System.out.println(obj.getName());

// there is no way to update the name after the object is created.

// obj.setName("new mkyong");

// System.out.println(obj.getName());

}

}

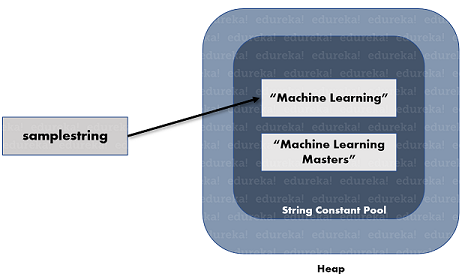
**Why are Strings Immutable in Java?**

Java uses the concepts of [string literals](https://www.edureka.co/blog/java-string/). So, if you consider an example where you have many reference variables referring a single object, then even if one reference variable changes the value of the [object](https://www.edureka.co/blog/java-object/), automatically all the other reference variables will be affected. Also, according to[Effective Java](https://www.oracle.com/technetwork/java/effectivejava-136174.html), chapter 4, page 73, 2nd edition, the following are reasons to use Immutable classes:

* Immutable objects are simple
* These objects require no synchronization and are inherently thread-safe
* Immutable objects make good building blocks for other objects

If I have to explain you with an example, then,

Let us say you have a variable ***samplestring***, which stores the string “**Machine Learning**“. Now, if you concatenate this string with another string *“****Masters****“,*then the object created for “**Machine Learning**” will not change. Instead, a new object will be created for “**Machine Learning Masters**“. Refer to the below image:



As you can see in the above image, the samplestring reference variable refers to “Machine Learning”, but not the other string, even after the creation of two objects. With this, we come to an end to this article on Mutable and Immutable in Java. I hope you guys are clear about each and every aspect that I have discussed above.

# **final vs Immutability in Java**

**final :**In Java, [final](https://www.geeksforgeeks.org/final-keyword-java/) is a modifier which is used for class, method and variable also. When a variable is declared with final keyword, it’s value can’t be modified, essentially, a constant.

[**Immutability**](https://www.geeksforgeeks.org/create-immutable-class-java/)**:**In simple terms, immutability means unchanging over time or unable to be changed. In Java, we know that String objects are immutable means we cant change anything to the existing String objects.

**Differences between final and immutability**

* final means that you can’t change the object’s reference to point to another reference or another object, but you can still mutate its state (using setter methods e.g). Whereas immutable means that the object’s actual value can’t be changed, but you can change its reference to another one.
* final modifier is applicable for variable but not for objects, Whereas immutability applicable for an object but not for variables.
* By declaring a reference variable as final, we won’t get any immutability nature, Even though reference variable is final. We can perform any type of change in the corresponding Object. But we cant perform reassignment for that variable.
* final ensures that the address of the object remains the same whereas the Immutable suggests that we can’t change the state of the object once created.

// Java program to illustrate

// difference between final

// and immutability

class Geeks {

    public static void main(String[] args)

    {

        final StringBuffer sb = new StringBuffer("Hello");

        // Even though reference varibale sb is final

        // We can perform any changes

        sb.append("GFG");

        System.out.println(sb);

        // Here we will get Compile time error

        // Because reassignment is not possible for final variable

        sb = new StringBuffer("Hello World");

        System.out.println(sb);

    }

}

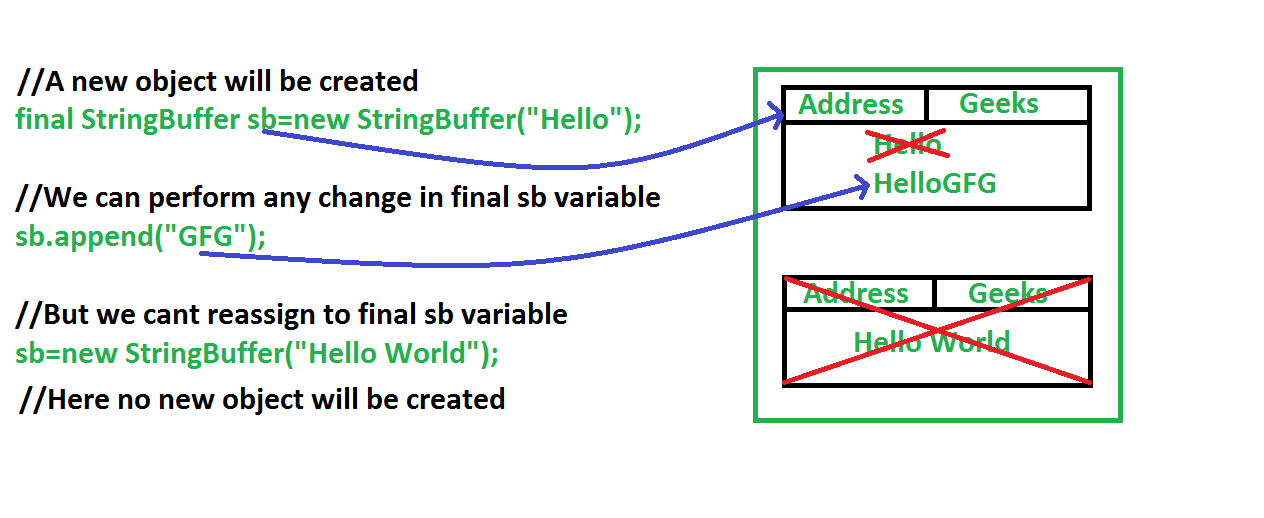
Geeks.java:14: error: cannot assign a value to final variable sb

sb = new StringBuffer("Hello World");

^

1 error

**Pictorial Representation of the above Program**



**Explanation:** In the above picture, we can see that we are creating an object of StringBuffer class by making reference final.

* Declaring reference variable as final, does not mean that the object is immutable.
* In the next line we are performing append() operation on the created object and it is successfully changed.
* If the object is immutable, then the above append operation can’t be done.
* But it is executed successfully as we declare reference variable as final. final means we can’t reassign anything to that reference variable again.
* Therefore when we try to create a new object of BufferedReader then it wont created any object by throwing an error to the console.

# **LinkedList in Java**

<https://www.geeksforgeeks.org/linked-list-in-java/>

# **Immutable List in Java**

<https://www.geeksforgeeks.org/immutable-list-in-java/?ref=lbp>

# **AbstractList in Java with Examples**

<https://www.geeksforgeeks.org/abstractlist-in-java-with-examples/?ref=lbp>

# **ArrayList in Java**

<https://www.geeksforgeeks.org/arraylist-in-java/?ref=lbp>

# **Constructors in Java**

<https://www.geeksforgeeks.org/constructors-in-java/>

# **Constructor Chaining In Java with Examples**

<https://www.geeksforgeeks.org/constructor-chaining-java-examples/>

# **Copy Constructor in Java**

<https://www.geeksforgeeks.org/copy-constructor-in-java/>

Java Constructors

<https://www.geeksforgeeks.org/java-gq/constructors-2-gq/>

# **Multithreading in Java**

<https://www.geeksforgeeks.org/multithreading-in-java/>

# **What does start() function do in multithreading in Java?**

<https://www.geeksforgeeks.org/start-function-multithreading-java/>

# **Deadlock in Java Multithreading**

<https://www.geeksforgeeks.org/deadlock-in-java-multithreading/>

# **Java Concurrency – yield(), sleep() and join() methods**

<https://www.geeksforgeeks.org/java-concurrency-yield-sleep-and-join-methods/?ref=lbp>

# **Inter-thread Communication in Java**

<https://www.geeksforgeeks.org/inter-thread-communication-java/?ref=lbp>

# **Synchronized in Java**

<https://www.geeksforgeeks.org/synchronized-in-java/?ref=lbp>

# **volatile-keyword-java**

<https://javarevisited.blogspot.com/2011/06/volatile-keyword-java-example-tutorial.html>

# **Lifecycle and States of a Thread in Java**

<https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/?ref=lbp>

# [**How can you implement**thread pool**executor.**](https://www.google.com/url?client=internal-element-cse&cx=009682134359037907028:tj6eafkv_be&q=https://practice.geeksforgeeks.org/problems/how-can-you-implement-thread-pool-executor&sa=U&ved=2ahUKEwi69rrXtYnqAhViyjgGHQLFDWAQFjACegQICBAC&usg=AOvVaw3dxroWce6UunMUzSzIOkZe)

<https://practice.geeksforgeeks.org/problems/how-can-you-implement-thread-pool-executor>

# **Thread Pools in Java**

<https://www.geeksforgeeks.org/thread-pools-java/>

**What is difference between different type of Thread pools**

<https://practice.geeksforgeeks.org/problems/what-is-difference-between-different-type-of-thread-pools>

# **Joining Threads in Java**

<https://www.geeksforgeeks.org/joining-threads-in-java/?ref=lbp>

# **Callable and Future in Java**

<https://www.geeksforgeeks.org/callable-future-java/?ref=lbp>

# **10 points about Thread in Java**

<https://www.java67.com/2014/01/10-points-about-thread-and-javalangthread-in-java.html>

# **Java Collections Interview Questions**

<https://www.javatpoint.com/java-collections-interview-questions>

# **Choosing the right Collection**

<http://www.javapractices.com/topic/TopicAction.do?Id=65>

# **Working of HashMap in Java**

<https://www.geeksforgeeks.org/internal-working-of-hashmap-java/>

<https://www.javatpoint.com/working-of-hashmap-in-java>

# **ConcurrentHashMap in java**

<https://www.geeksforgeeks.org/concurrenthashmap-in-java/>

<https://java2blog.com/concurrenthashmap-in-java/>

# **Difference between HashMap and ConcurrentHashMap**

<https://www.geeksforgeeks.org/difference-hashmap-concurrenthashmap/>

# **Difference between Comparable and Comparator**

<https://www.javatpoint.com/difference-between-comparable-and-comparator>

# **Core Java - OOPs Concepts: Inheritance Interview Questions**

<https://www.javatpoint.com/corejava-interview-questions#corebasicsinterview>

Questions starts from 51

# **Garbage Collection in Java**

<https://www.geeksforgeeks.org/garbage-collection-java/>

<https://www.java67.com/2020/02/50-garbage-collection-interview-questions-answers-java.html>

# **Types of JVM Garbage Collectors in Java with implementation details**

<https://www.geeksforgeeks.org/types-of-jvm-garbage-collectors-in-java-with-implementation-details/>

# **How to make object eligible for garbage collection in Java?**

<https://www.geeksforgeeks.org/how-to-make-object-eligible-for-garbage-collection/>

# **How to prevent objects of a class from Garbage Collection in Java**

<https://www.geeksforgeeks.org/how-to-prevent-objects-of-a-class-from-garbage-collection-in-java/>

# **Introduction to Data Structures | 10 most commonly used Data Structures**

<https://www.geeksforgeeks.org/introduction-to-data-structures-10-most-commonly-used-data-structures/>

# **Sorting in Java**

<https://www.geeksforgeeks.org/sorting-in-java/>

# **Data Structure and Algorithm**

<https://www.java67.com/2018/06/data-structure-and-algorithm-interview-questions-programmers.html>

<https://javarevisited.blogspot.com/2013/03/top-15-data-structures-algorithm-interview-questions-answers-java-programming.html>

Sorting Algorithms

<https://www.geeksforgeeks.org/sorting-algorithms/>

# **Time Complexities of all Sorting Algorithms**

<https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/>

# **Comparison among Bubble Sort, Selection Sort and Insertion Sort**

<https://www.geeksforgeeks.org/comparison-among-bubble-sort-selection-sort-and-insertion-sort/>

# **Binary Search**

<https://www.geeksforgeeks.org/binary-search/>

# **Binary Search in Java**

<https://www.geeksforgeeks.org/binary-search-in-java/>

# **Stack vs Queue in Java**

<https://javarevisited.blogspot.com/2017/03/difference-between-stack-and-queue-data-structure-in-java.html#axzz56fRo1HGs>

# **Spring vs. Spring Boot vs. Spring MVC**

<https://www.javatpoint.com/spring-vs-spring-boot-vs-spring-mvc>

# **Top Spring Interview Questions**

<https://codingnconcepts.com/top-spring-interview-questions/>

<https://www.softwaretestinghelp.com/spring-interview-questions/>

# **Spring Boot**

<https://www.javatpoint.com/spring-boot-tutorial>

<https://codingnconcepts.com/top-spring-boot-interview-questions/>

<https://www.knowledgehut.com/interview-questions/spring-boot>

# **Introduction to RESTful Web Services**

<https://www.javatpoint.com/restful-web-services-spring-boot>

# **Spring Boot Starter Actuator**

<https://www.javatpoint.com/spring-boot-actuator>

# **Monolithic vs Microservices architecture**

<https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/>

# **Hibernate**

<https://www.decodejava.com/hibernate-value-type-object.htm>

# **Heap And stack**

<https://www.journaldev.com/4098/java-heap-space-vs-stack-memory>

# [Difference Between Collections And Streams In Java](https://javaconceptoftheday.com/collections-and-streams-in-java/)

<https://javaconceptoftheday.com/collections-and-streams-in-java/>

# [Interface Vs Abstract Class After Java 8](https://javaconceptoftheday.com/interface-vs-abstract-class-after-java-8/)

<https://javaconceptoftheday.com/interface-vs-abstract-class-after-java-8/>

# **Top Answers to Java Interview Questions**

<https://www.goeduhub.com/1775/java-interview-questions-set-1>

<https://www.goeduhub.com/1825/java-interview-questions-set-2>

<https://intellipaat.com/blog/interview-question/java-interview-questions/>

<https://codingnconcepts.com/top-java-interview-questions/>

https://www.java67.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html

https://www.java67.com/2012/09/top-10-tough-core-java-interview-questions-answers.html

https://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html

<https://javarevisited.blogspot.com/2018/02/top-20-spring-rest-interview-questions-answers-java.html>

<https://www.knowledgehut.com/interview-questions/java>

<https://www.knowledgehut.com/interview-questions/java-collection>

<https://www.knowledgehut.com/interview-questions/hibernate>

<https://javarevisited.blogspot.com/2011/04/top-20-core-java-interview-questions.html#axzz6PvNpNiGj>

<https://javarevisited.wordpress.com/2011/07/02/java-interview-questions-for-senior-java-developers/>

# **Java Design Pattern Interview Questions and Answers**

<https://www.javatpoint.com/java-design-pattern-interview-questions>

<https://www.java67.com/2012/09/top-10-java-design-pattern-interview-question-answer.html>

<https://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html>

<https://medium.com/@kevalpatel2106/how-to-make-the-perfect-singleton-de6b951dfdb0>

<https://j2eereference.com/singleton-in-clustered-environment/>

<https://www.java-success.com/?gclid=Cj0KCQjwirz3BRD_ARIsAImf7LMb_LzLlZws_6M3oqw5r1ShAPyKI2gyjS31t9AvYpNMtFPfBM7rppIaApvGEALw_wcB>

<https://www.java-success.com/core-java-interview-questions/>

<https://www.java-success.com/proxy-decorator-adapter-bridge-facade-design-patterns-look-similar-differences/>

<https://www.java-success.com/java-garbage-collection-interview-questions-and-answers/>

<https://www.java-success.com/java-multithreading-interview-questions-and-answers/>

<https://www.java-success.com/advanced-java-interview-questions-answers-to-judge-experience/>

<https://www.java-success.com/java-futuretask-example/>

<https://www.java-success.com/10-%e2%99%a6-executorservice-vs-forkjoin-future-vs-completablefuture-interview-qa/>

<https://www.java-success.com/producer-and-consumer-java-multi-threading-code/>

<https://www.interviewsansar.com/category/java-interview-questions-and-answers/>

<https://www.interviewsansar.com/category/java-collections-interview-questions-and-answers/>

<https://www.interviewsansar.com/category/java-multithreading-interview-questions-and-answers/>

<https://www.javamadesoeasy.com/2015/03/threads-top-75-interview-questions-and_9.html>

# JOB Freelancer at Toptal

<https://www.toptal.com/java/interview-questions>

# MCQ online

<https://javasearch.buggybread.com/OnlineTest/javaTest.php>