## StringBuffer

⇒ StringBuffer is a pre-defined class present in java.lang package
⇒ Syntax :-

```
public final class StringBuffer
                        extends AbstractStringBuilder
                        implements Appendable, Serializable, Comparable<StringBuffer>, CharSequence
{
        // constructors
        // methods
}
```

⇒ String objects created by StringBuffer are mutable
⇒ All the methods of StringBuffer are "synchronized"

## StringBuilder

⇒ StringBuilder is also a pre-defined class present in java.util package
⇒ Syntax :-

```
public final class StringBuilder
                    extends AbstractStringBuilder
                    implements Appendable, Serializable, Comparable<StringBuilder>, CharSequence
{
    // constructors
    // methods
}
```

⇒ String objects created by StringBuilder are mutable
⇒ All the methods of StringBuilder are "non-synchronized"

## Difference between StringBuffer & StringBuilder

⇒ StringBuffer are synchronized
StringBuilder are non-synchronized

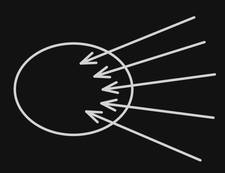
⇒ StringBuffer are slow (low performance)
StringBuilder are fast (high performance)

⇒ StringBuffer follows the sequential execution
StringBuilder follows the parallel execution

⇒ StringBuffer provide the guarantee for data consistency
StringBuilder does not provide guarantee for data consistency

⇒ StringBuffer is thread-safe
StringBuilder is not thread-safe

⇒ StringBuffer came in JDK 1.0 version
StringBuilder came in JDK 1.5 version

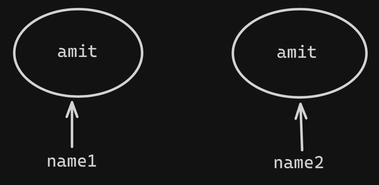


## Difference between String and StringBuffer

⇒ String are immutable
StringBuffer are mutable

⇒ String methods are non-syncronized
StringBuffer methods are synchronized

## Difference between .equals() and == operator




name1.equals(name2) → true

name1 = = name2 → false

.equals method compares the string content
== operator compares the reference

.equals method is not String class method, it overrides the Object class .equals method

```
class Object
{
    equals()
    {
        compare -> reference
    }
}

class String extends Object
{
    @Override
    equals()
    {
        compare -> content
    }
}
```

## Interview Questions

⇒ StringBuffer
1. What is StringBuffer in Java?
2. Why do we need StringBuffer if we already have String?
3. Is StringBuffer mutable or immutable?
4. Is StringBuffer thread-safe? How is synchronization handled?
5. What happens if multiple threads access a StringBuffer object simultaneously?
6. What are some commonly used methods in StringBuffer?
7. What is the default capacity of a StringBuffer? How does it increase internally?
8. What is the difference between length() and capacity() methods in StringBuffer?
9. How does the ensureCapacity() method work in StringBuffer?

⇒ StringBuilder
10. What is StringBuilder in Java?
11. Why was StringBuilder introduced in Java 5?
12, Is StringBuilder mutable or immutable?
13. Is StringBuilder thread-safe? Why or why not?
14. What are some commonly used methods in StringBuilder?
15. What is the default capacity of a StringBuilder? How does it expand?
16. What is the role of trimToSize() in StringBuilder?

⇒ Comparison: String vs StringBuffer vs StringBuilder
17. What is the difference between String, StringBuffer, and StringBuilder?
18. When should we use StringBuffer instead of String?
19. When should we use StringBuilder instead of StringBuffer?
20. Which is faster: StringBuffer or StringBuilder? Why?
21. Which one is preferable for multi-threaded applications?
22. Which one is preferable for single-threaded applications?
23. Can we convert a StringBuffer or StringBuilder to a String? How?
24. Can we convert a String to a StringBuffer or StringBuilder? How?

⇒ Methods & Internals
25. How do append() and insert() methods work in StringBuffer/StringBuilder?
26. What is the difference between delete(), deleteCharAt(), and clear() operations?
27. How does the reverse() method work?
28. What is the difference between setLength() and substring() methods?
29. How does replace() work in StringBuffer/StringBuilder?
30. What happens if we exceed the current capacity of a StringBuffer/StringBuilder?
31. How does memory allocation differ between String and StringBuffer/StringBuilder?
32. Can we override methods of StringBuffer or StringBuilder? Why or why not?