Learning Goals

Click to Continue

This session will help you to understand,

- What is String and String buffer?
- Writing a program to explain usage of string and string buffer?

String

Click to Continue

Strings, *are a sequence of characters*. ***String*** class is used for creating and processing strings. Strings are ***objects***.

Creating Strings:

Option 1: String *message* = "Hello!"; // Creates a string literal and assign it to a String reference.

Option 2: String *message* = new String("Hello!"); // String object created using constructor.

Salient Points about String

Click to Continue

String class is present in **java.lang** package.

Few String API's:

- Searching strings – These API's are used to search a character or a string inside a string.
- Extracting substrings – These API's are used to extract a substring from a string
- Comparing two Strings - These API's are used to compare two strings.

Try it out -String Constructors

Click to Continue

Develop a class StringDemo with a main method which performs a following operations,

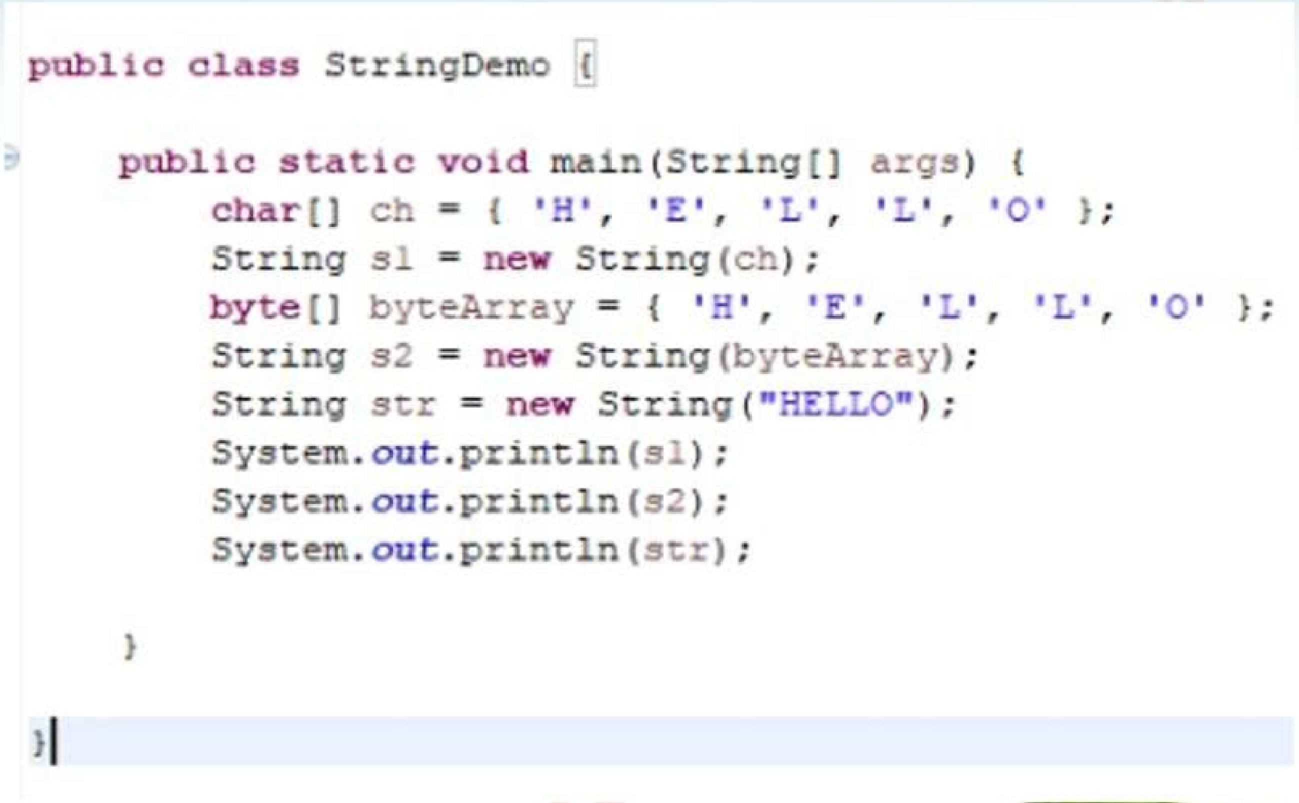
- Create a String (S1) from a char array ('H','E','L','L','O'}
- Create a String from a byte array ('H','E','L','L','O'}
- Create a empty String using one of the constructor.
- Create a String object from the string S1 created in step 1.
- Print all the Strings created in the above points using SOP.

Try it out -String Constructors

Click to Continue

Develop a class StringDemo with a main method which performs a following operations,

- Create a String (S1) from a char array ('H','E','L','L','O'}
- Create a String from a byte array ('H','E','L','L','O'}
- Create a empty String using one of the constructor.
- Create a String object from the string S1 created in step 1.
- Print all the Strings created in the above points using SOP.



```
public class StringDemo {

    public static void main(String[] args) {
        char[] ch = { 'H', 'E', 'L', 'L', 'O' };
        String s1 = new String(ch);
        byte[] byteArray = { 'H', 'E', 'L', 'L', 'O' };
        String s2 = new String(byteArray);
        String str = new String("HELLO");
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(str);

    }

}
```

Few String API's

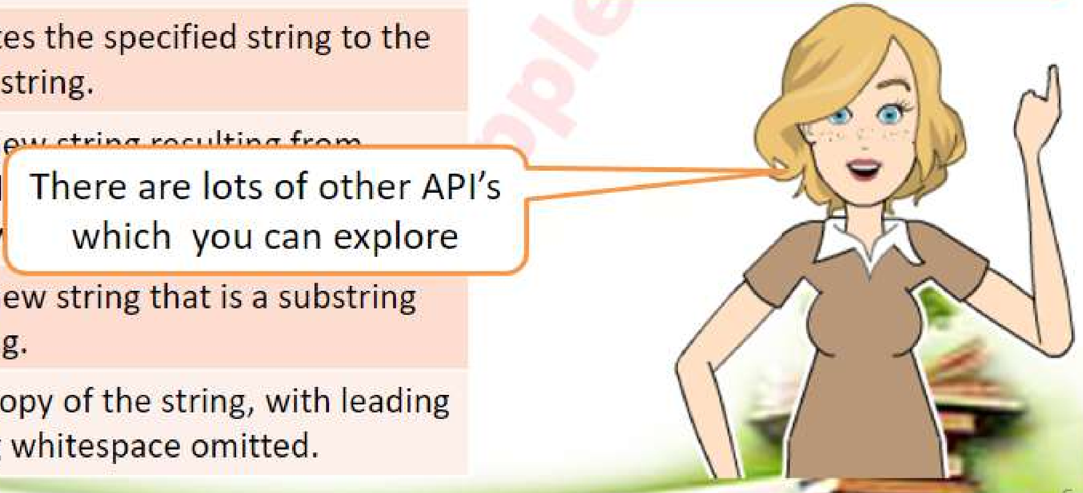
Click to Continue

Result	Method	Description
char	charAt(int index)	Returns the character at the specified index.
boolean	equalsIgnoreCase(String anotherString)	Compares this String to another String, ignoring case considerations.
int	indexOf(int ch)	Returns the index within this string of the first occurrence of the specified character.
int	length()	Returns the length of this string.
String	concat(String str)	Concatenates the specified string to the end of this string.
String	replace(char oldChar, char newChar)	Returns a new string resulting from replacing al this string v
String	substring(int beginIndex, int endIndex)	Returns a new string that is a substring of this string.
String	trim()	Returns a copy of the string, with leading and trailing whitespace omitted.

Returns a character in the specified index.

Compares two strings whether it is equal or not.

This returns the length of the string.



String are immutable

Click to Continue

Strings are immutable once created a String object cannot be changed.

Illustration:

```
String S1="One";
S1="Two";
```

Value of S1 will be "Two".

String are immutable

Click to Continue

Strings are immutable once created a String object cannot be changed.

Illustration:

```
String S1="One";
S1="Two";
```

Value of S1 will be "Two".

String S1 will have a address A1 and assign the value One.

String are immutable

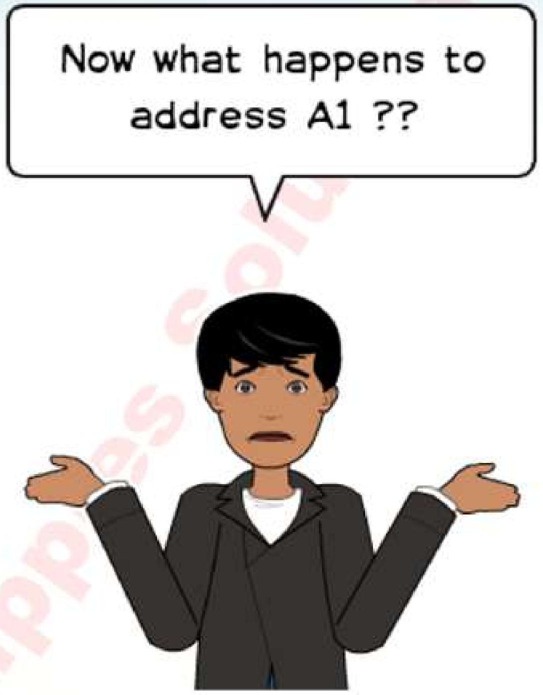
Click to Continue

Strings are immutable once created a String object cannot be changed.

Illustration:

```
String S1="One";
S1="Two";
```

Value of S1 will be "Two".



String are immutable

Click to Continue

Strings are immutable once created a String object cannot be changed.

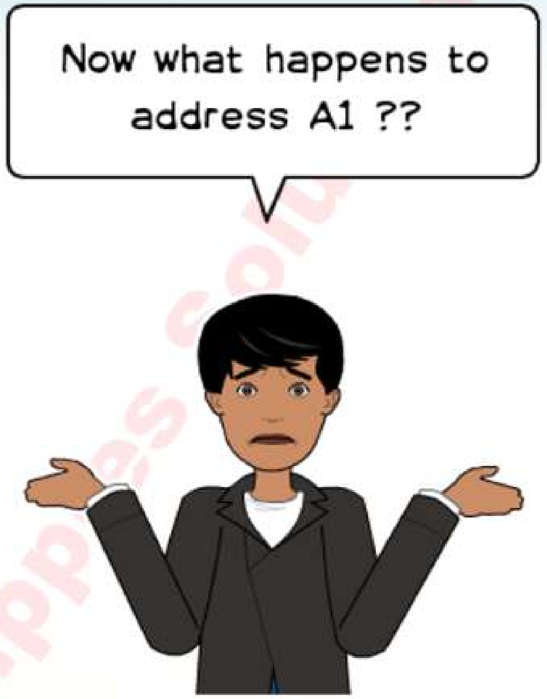
Illustration:

```
String S1="One";
S1="Two";
```

Value of S1 will be "Two".

Address A1 will be removed from the memory by the JVM. Since the string is not editable in the original address A1 String is called immutable.

NOTE: All string operations (viz trim, substring) returns a new String Object without changing the original String.

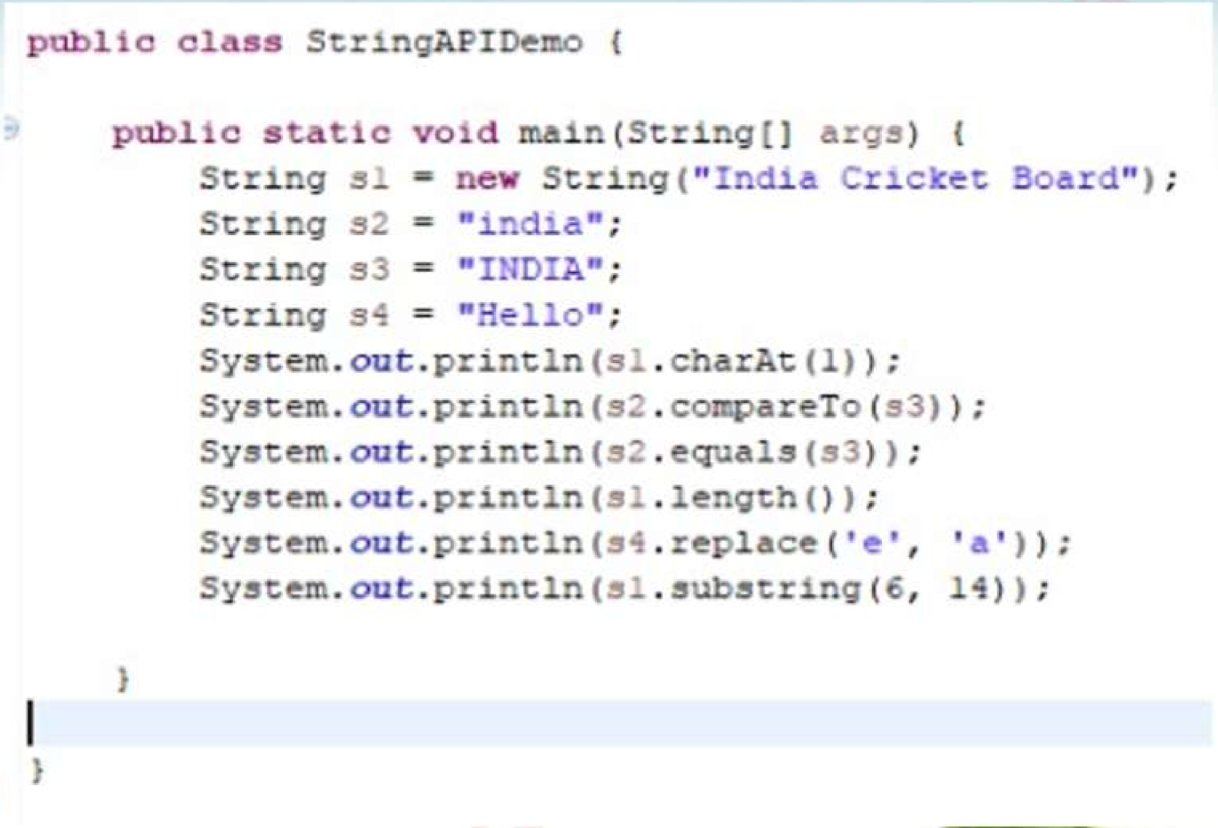


Try it out -String API's

Click to Continue

Develop a class StringAPIDemo with a main method which performs a following operations,

- For a given String "India Cricket Board" print the second character.
- Create two Strings "India" & "INDIA" and compare them with an API and print the result.
- For the above two strings check whether they are equal and print the result.
- For a String "India Cricket Board" print the length of the String.
- In a String "Hello" replace the letter 'e' with 'a' and print the same.
- For a given String "India Cricket Board" extract the string "Cricket" and print the same.



```java
public class StringAPIDemo {

    public static void main(String[] args) {
        String s1 = new String("India Cricket Board");
        String s2 = "india";
        String s3 = "INDIA";
        String s4 = "Hello";
        System.out.println(s1.charAt(1));
        System.out.println(s2.compareTo(s3));
        System.out.println(s2.equals(s3));
        System.out.println(s1.length());
        System.out.println(s4.replace('e', 'a'));
        System.out.println(s1.substring(6, 14));

    }

}
```

String Buffer

Click to Continue

StringBuffer are ***mutable*** String, that is string value can be changed in the original address.

- The String buffer API can be used to change the length and content of String without creating a new object..
- All API's of StringBuffer are ***synchronized***.
- This is recommended if String objects value needs to be changed.

String Buffer usage:

- Concatenating two String
- Changing string value by inserting characters in the string.
- Deleting few characters from the String.

String Buffer Constructors

Click to Continue

Constructors to create ***StringBuffer***

- **Method I** - StringBuffer buffer = new StringBuffer();
- **Method II** - StringBuffer buffer = new StringBuffer(“Alan”);
- **Method III** - StringBuffer buffer = new StringBuffer(30);

creates a String Buffer with a number

String Buffer to concatenate strings

Click to Continue

How is a string concatenated?

```
String S="Hello";
s = s+ " World";
```

As seen earlier this creates a new address every time a string is concatenated. This is not the efficient way.

Use String buffer for concatenation

This appends the string "World" with hello in address A1. This will not create a new address.

Since string buffer changes the value in the original address A1, ***it is referred to as mutable***.

```
s = new StringBuffer("Hello").append("World").toString();
```

This creates a buffer in the address A1 with string Hello

Finally convert the buffer to a string.

StringBuffer API

Click to Continue

Result	Method	Description
StringBuffer	insert(int offset, String str)	Inserts the string argument into this string buffer.
StringBuffer	delete(int start, int end)	Removes the characters in a substring of this StringBuffer.
StringBuffer	replace(int start, int end, String str)	Replaces the characters in a substring of this StringBuffer with characters in the specified String.
StringBuffer	reverse()	The character sequence contained in this string buffer is replaced by the reverse of the sequence.
StringBuffer	append(String str)	Appends the string to buffer.

Inserts a string into another string

Deletes a string from the original string based on the index passed

Reverses a string

There are lots of other API's which you can explore

Try It Out – StringBuffer

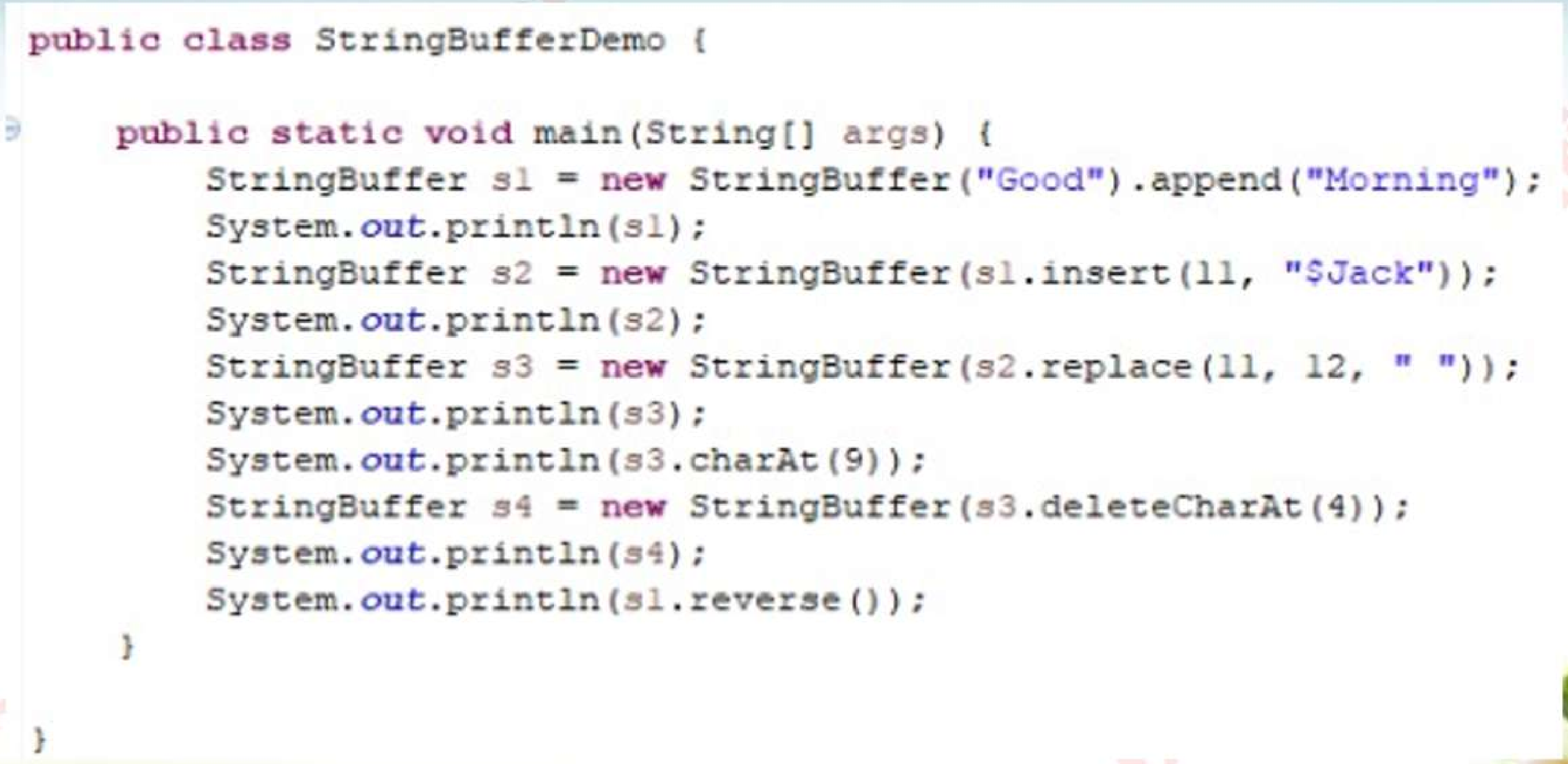
Click to Continue

Develop a class StringBufferDemo with a main method performing the below logic,

1. Append two Strings “Good” & “Morning” without using “+” operator. Store it in a StringBuffer variable S1 and print it. Result: “Good Morning”
2. Insert a string “$Jack” in the String S1 after “Morning”. Store it in a StringBuffer variable S2 and print it. Result: “Good Morning$Jack”
3. Replace $ with space in S2. Store it in a StringBuffer variable S3 and print it. Result: “Good Morning Jack”.
4. Print the character at the 9’th position of String S3. Result: n
5. Delete the character in the 6 position in String S3. Store it in a StringBuffer variable S4 and print it.. Result: “Good orning Jack”
6. Reverse the string S1 and print the string. Result: Good morning will be printed in the reverse order

Try it out - Solution

Click to Continue



```
public class StringBufferDemo {

    public static void main(String[] args) {
        StringBuffer s1 = new StringBuffer("Good").append("Morning");
        System.out.println(s1);
        StringBuffer s2 = new StringBuffer(s1.insert(11, "$Jack"));
        System.out.println(s2);
        StringBuffer s3 = new StringBuffer(s2.replace(11, 12, " "));
        System.out.println(s3);
        System.out.println(s3.charAt(9));
        StringBuffer s4 = new StringBuffer(s3.deleteCharAt(4));
        System.out.println(s4);
        System.out.println(s1.reverse());
    }

}
```