Chapter 6
Java Strings

The division of the chapter is as follows:

Chap 6 Java Strings

Introduction:

- In Java, strings are handled by String, StringBuffer and StringBuilder classes.
- The java.lang.Object class in the parent class of these classes.

String Class:

- In Java, there is no primitive type to store the string data.
- To handle string data in Java, we require objects of the String class.
- String class is included in the java.lang package. Therefore, each class can use the String class without importing any package.
- String objects are immutable, that means the value of the objects can never be changed.

String Concatenation using + operator :

- Java allows operator + to be used with String objects which results into concatenation of two strings producing a String object as a result.

```
class SD2 {
        public static void main(String args[ ]) {
                String name = "Akshay";
                String str="Name is  "+name+" Kumar";
                System.out.println(str);
        }}
```

Output: Name is Akshay Kumar

String Concatenation using += operator :

- Java allows operator += to be used with String objects which results into concatenation of two strings producing a String object as a result.

```
class SD2 {
        public static void main(String args[ ]) {
                String str = "Akshay";
                str += "Kumar";
                System.out.println(str);
        }}
```

Output: Akshay Kumar

Note: With string, + , += == != can be used

If we try to use any other Operator we get compilation error.

String Methods :

length()		
1	String Function	int length()
	Description	➢ The length of String is the number of characters present in it. ➢ The method to calculate the length of string is length().
	Eg:	String str = "INDIA"; System.out.println(str); System.out.println(str.length());
	Output:	INDIA 5

P)WAJP to

1) **Read two strings from the user.**
2) **Print the length of two strings.**
3) **Concatenate two strings and display the result.**

Program:

```
import java.io.*;
class StrProg1 {
public static void main(String args[ ]) throws IOException {
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String str1, str2, str3;
int len1, len2;

System.out.println("Enter First String : " );
str1 = br.readLine();
System.out.println("Enter Second String : "
str2 = br.readLine();

len1= str1.length();
System.out.println("First String : " + str1 + " length is " + len1);
len2= str2.length();
System.out.println("Second String : " + str2 + " length is " +
len2);

str3 = str1+" is "+str2;
System.out.println("Concatenated String : " + str3 );
}
}
```

Output:

```
>javac StrProg1.java

>java StrProg1
Enter First String :
Java
Enter Second String :
Secured and Portable
First String : Java length is 4
Second String : Secured and
Portable length is 20
Concatenated String : Java is
Secured and Portable
```

Strings are immutable.

substring()

2	String Function	String substring(int start_index) String substring(int start_index, int end_index)
	Description	➢ The first form extracts a substring from invoking string object from start_index to the end of the string and returns it ➢ The second form extracts a substring from invoking string object from start_index to the end_index-1 and returns it

	Eg:	String s="Java is Kool";
		String temp;
		System.out.println(s);
		temp = s.substring(0);
		System.out.println(temp);
		temp = s.substring(5);
		System.out.println(temp);
		temp = s.substring(0,3);
		System.out.println(temp);
		0 to 2 →

	Output:	Java is Kool Java is Kool Jav

Note: if the index range is not proper then JVM throws
StringIndexOutOfBoundsException.

concat()

3	String Function	String concat(String str)
	Description	➢ The str is concatenated with the invoking object's string and the resultant string is returned.

	Eg:	String s1 ="Abdul";
		String s2 =" Kalam";
		String temp;
		System.out.println(s1);
		System.out.println(s2);
		temp = s1.concat(s2);
		System.out.println(temp);
		temp= s1 + s2;
		System.out.println(temp);

	Output:	Abdul Kalam Abdul Kalam Abdul Kalam

Note: if s1 or s2 is declared but not initialized in method then compiler throws an error: variable s1 might not have been initialized.

replace(), replaceFirst() & replaceAll()

4	String Function	String replace(char original, char replacement) String replaceFirst(String original_str, String new_str) String replaceAll(String original_str, String new_str)
	Description	➢ The original character is replaced by the replacement character. ➢ the method will return the String object with the first occurrence of the original_str within invoking String object being replaced by new_str. ➢ the method will return the String object with all occurrence of the original_str within invoking String object being replaced by new_str.
	Eg:	String s1 = "Bye Bye Bye !!"; String s2= s1. replace('y','e'); System.out.println(s1); System.out.println(s2); String s3= s1.replace("By","e"); System.out.println(s1); System.out.println(s3); String s4= s1.replaceFirst("By","Me"); System.out.println(s1); System.out.println(s4); String s5= s1.replaceAll("By","Me"); System.out.println(s1); System.out.println(s5);
	Output:	Bye Bye Bye!! Bee Bee Bee !! Bye Bye Bye!! eee ee ee!! Bye Bye Bye!! Mee Bye Bye!! Bye Bye Bye!! Mee Mee Mee!!

Note1: replace() Can work with char and String

Note2: replaceFirst() and replaceAll() work with strings

Note3: If the original (c/s) is not present then nothing would be replaced.

P) WAJP to replace a substring within a string by any other string .

Program:

```
import java.io.*;
class StrProg2 {
public static void main(String args[ ]) throws IOException  {

InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String str1, str2, str3, str4;

System.out.println("Enter the  String : " );
str1 = br.readLine();
System.out.println("Enter the substring : " );
str2 = br.readLine();
System.out.println("Enter the new substring : " );
str3 = br.readLine();

str4 = Str1.replace(Str2, Str3);

System.out.println("Original  String :   " + str1 );
System.out.println("Modified String :   " + str4 );
}
}
```

Output:

```
>java StrProg2
Enter the String
this is kool
Enter the Substring
is
Enter the new substring
ay
Original String : this is kool
Modified String : thay ay kool
```

Trimming and Changing Case

5	String Function	String trim() String toLowerCase() String toUpperCase()
	Description	➢ the method trim() is used to remove leading and trailing blank white spaces. ➢ method toLowerCase() converts all the characters in a string from uppercase to lowercase. ➢ method toUpperCase() converts all the characters in a string from lowercase to uppercase. ➢ The non-alphabetical characters such as digits, blank spaces, punctuation marks etc. remains unaffected.

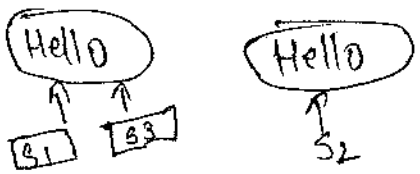
	Eg:	String s1 = " Mahatma Gandhi ";
		String s2 = "MaHATMA";
		String s3 = "gandhi";
		String temp;
		
		temp = s1.trim();
		System.out.println(s1);
		System.out.println(temp);
		temp=s2.toLowerCase();
		System.out.println(s2);
		System.out.println(temp);
		System.out.println(s3);
		temp=s3.toUpperCase();
		System.out.println(temp);

	Output:	Mahatma Gandhi Mahatma Gandhi maHATMA mahatma gandhi GANDHI

Comparision

6	String Function	boolean equals() boolean equalsIgnoreCase()
	Description	➢ the method returns true if both these strings are exactly same otherwise it returns false. ➢ the comparison of the two strings is case-sensitive. ➢ equalsIgnoreCase() is case-insensitive.
	Eg1:	String s1 = "Hello"; String s2 = new String("Hello"); String s3 = "Hello"; if (s1 == s2) = = checks whether ref. to same obj. System.out.println("Both are same (==) ");

Heap Memory:




```
else
    System.out.println("Both are not same (==) ");
if (s1.equals(s2))  equals checks the content
    System.out.println("Both are same (equals) ");
else
    System.out.println("Both are not same (equals) ");
if (s1 == s3)
    System.out.println("Both are same (==) ");
else
    System.out.println("Both are not same (==) ");
if (s1.equals(s3))
    System.out.println("Both are same (equals) ");
else
    System.out.println("Both are not same (equals) ");
```

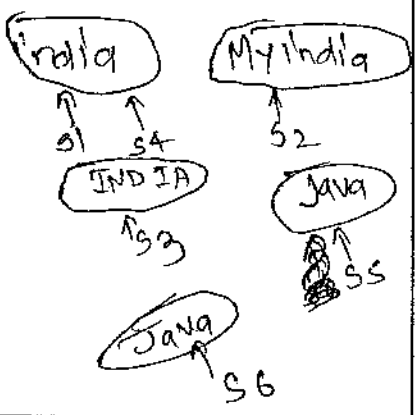
Output:

Both are not same (==)
Both are same (equals)
Both are same (==)
Both are same (equals)

Note: equals() checks for the contents of the objects
and == checks for the references made to the objects.

Eg2:

Heap Memory:




```
String s1="India";
String s2=" My India";
String s3="INDIA";
String s4="India";
System.out.println(s1.equals(s4));
System.out.println(s1.equals(s2));
System.out.println(s1.equals(s3));
System.out.println(s1.equalsIgnoreCase(s3));
String s5="Java";
String s6=new String(s5);
System.out.println(s5.equals(s6));
System.out.println(s5==s6);
System.out.println(s5==s5);
```

Output:

True
False
false
True
True
False
True

Starting and Ending with

7	String Function	boolean startsWith() boolean endsWith()
	Description	➢ startsWith() :It determines whether a given string begins with a specified string. ➢ endsWith() : It determines whether a given string ends with a specified string.
	Eg:	String s1="Yahoo"; System.out.println(s1.startsWith("Ya")); System.out.println(s1.endsWith("hoo"));
	Output:	True ~~False~~ True

Searching Strings

Note: If character is not found then the method returns -1

8	String Function	int indexOf() int lastindexOf()
	Description	➢ indexOf() : searches for the first occurrence of a character or substring ➢ lastIndexOf() : searches for the last occurrence of a character or substring
	Eg:	String s = "Java is Easy"; System.out.println(s.indexOf('a')); System.out.println(s.lastIndexOf('a'));
	Output:	1 9

Java is Easy
0 1 2

Character Extraction

9	String Function	char charAt(int position)
	Description	➢ The charAt() will obtain a character from the position and will return that character.
	Eg:	String s = "INDIA"; char ch; System.out.println(s); ch= s.charAt(2); System.out.println(ch);
	Output:	INDIA D

P3)WAJP to
1) Compare two strings
2) Concatenate them if they are equal.
3) The concatenated string should be displayed in lower case.

Program:

```
import java.io.*;

class StrProg3 {
public static void main(String args[ ]) throws IOException  {

InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String str1, str2, str3, str4;
boolean result;

System.out.println("Enter the  First String : " );
str1 = br.readLine();
System.out.println("Enter the Second string : " );
str2 = br.readLine();
result = Str1.equalsIsIgnoreCase(str2);
if(result){
    str3 = str1.concat(str2);
    str4 = str3.toLowerCase();
    SOP("Concatenated String : "+str4);
}
else
SOP("Strings are not same");
}
}
```

Output:

```
>javac StrProg3.java
>java StrProg3
Enter the First string:
Java
Enter the Second string:
Java
Concatenated String:
javajava.
```

StringBuffer & StringBuilder:

StringBuffer and StringBuilder Class: ➔ Similarities

1. They are mutable

2. Same APIs.

StringBuffer and StringBuilder Class: ➔ Differences

No.	StringBuffer	StringBuilder
1	Methods are thread-safe	Methods are not thread-safe.
2 (one at a time)	Methods are synchronized and hence slow.	not synchronized hence fast
3	Can be used in both STE and MTE	Can be used only in STE.
4	available from java 1.0	Available from java 5.0

mutual exclusion

race condition

Use of StringBuffer and StringBuilder:

- StringBuffer and StringBuilder classes are commonly used to perform IO operation when large streams of input are being handled by the program.
- in such cases, large blocks of characters are handled as units, and the StringBuffer objects are ideal to handle a block of data, pass it on and reuse the memory to handle the next block of data.

Very Imp. In String we have to assign to it. No case in StringBuffer

String Object	StringBuffer Object

String s = new String("Hello");
s.concat(" World");
 	
The above will show the output as: Hello
 	
s = s.concat(" World");
The above will show the output as: Hello World.
	StringBuffer sb = new StringBuffer("Hello");
	sb.append(" World");
	
	The above will show the output as: Hello World.

Note: String has concat() method And StringBuffer has append() Method.

StringBuffer Constructors:

- **StringBuffer( )**
  - the default constructor reserves room for 16 characters without reallocation.
- **StringBuffer(int size)**
  - accepts an integer that explicitly sets the size of the buffer.
- **StringBuffer (String str)**
  - accepts a string argument that sets the initial contents of the StringBuffer objects and reserves room for 16 more characters.

StringBuffer Methods:

length() and capacity()		
1	String Function	int length() int capacity()
	Description	int length() → It returns the current length of StringBuffer int capacity() → It returns the total allocated capacity
	Eg:	StringBuffer sb= new StringBuffer("Java"); System.out.printin(sb); System.out.println(sb.length()); System.out.println(sb.capacity());
	Output:	Java 4 20 Java 0123456 . . . . 19 It is predefined by java will add more to 20

charAt() and setCharAt()

2	String Function	char charAt(int where) void setCharAt(int where, char ch)
	Description	char charAt(int where) → It returns the character at that position where in the StringBuffer. void setCharAt(int where, char ch) → This will set the character within the StringBuffer.
	Eg:	StringBuffer sb = new StringBuffer("Java"); System.out.println(sb); System.out.println(sb.charAt(0)); sb.setCharAt(0,'K'); System.out.println(sb); System.out.println(sb.charAt(0));
	Output:	Java J Kava

append()

3	String Function	append()
	Description	StringBuffer append() → It concatenates the string with any any other types of data to the end.
	Eg:	StringBuffer sb = new StringBuffer(); sb.append(" X = 100 "); System.out.println(sb);
	Output:	X = 100

insert()

4	String Function	insert()
	Description	StringBuffer insert()→ It inserts one string into another.
	Eg:	StringBuffer sb = new StringBuffer("Java is Kool"); sb.insert(8, "super-"); System.out.println(sb);
	Output:	Java is Super-Kool.

reverse()

5	String Function	reverse()
	Description	StringBuffer reverse()→ It reverses the characters within StringBuffer
	Eg:	StringBuffer sb = new StringBuffer("JAVA"); System.out.println(sb); sb.reverse(); System.out.println(sb);
	Output:	AVAJ JAVA AVAJ

delete() and deleteCharAt()

6	String Function	delete() deleteCharAt()
	Description	StringBuffer delete(int startindex, int endindex) → It deletes a sequence of characters from the StringBuffer from the startindex to endindex - 1. StringBuffer deleteCharAt(int loc) →It deletes the character at the index specified by loc.

	Eg:	StringBuffer sb = new StringBuffer("Java is Kool");
		0 1 2 3 4 5 6 7
		System.out.println(sb);
		sb.delete(2 ,5);
		System.out.println(sb);
		sb.deleteCharAt(0);
		System.out.println(sb);

	Output:	Jais Kool ais Kool

replace()

7	String Function	replace()
	Description	StringBuffer replace(int startindex, int endindex, String str) → It replaces one set of characters with another set StringBuffer
	Eg:	StringBuffer sb = new StringBuffer("He is Superstar."); System.out.println(sb); sb.replace(3,5,"was"); → i.e 3 to 4. System.out.println(sb);
	Output:	He is Superstar 0 1 2 3 4 5 6 He wassuperstar space.

substring()

8	String Function	String substring()
	Description	StringBuffer substring(int startindex, int endindex) → It returns the substring StringBuffer
	Eg:	StringBuffer sb = new StringBuffer("He is Superstar."); String s; System.out.println(sb); s = sb.substring(6,11); System.out.println(s);
	Output:	He is Superstar Super.

Chained Methods:

- Java supports the concept called chained methods, in which one method can be chained to another method, to another method and so on.
- Chained method has the general form of :
  result = method1().method2().method3();
- How to decipher this:
  - Determine what the leftmost method call will return (say call it x).
  - Use x as the object invoking the second (from the left) method. (say call it y).
  - Use y as the object invoking the third (from the left) method ...

Eg1:

```
class SbCh1
{
public static void main(String args[])
{
StringBuilder sb = new StringBuilder("abc");
sb.append("def").reverse().insert(3, "--");
System.out.println(sb);
}
}
```

Output

fed--cba

abc

0	1	2	3	4	5	6	7
a	b	c					

append("def") → abcdef

0	1	2	3	4	5	6	7
a	b	c	d	e	f		

reverse() → fedcba

0	1	2	3	4	5	6	7
f	e	d	c	b	a		

insert(3, "--") → fed--cba

0	1	2	3	4	5	6	7
f	e	d	-	-	c	b	a

Eg2:

```
class SbCh2
{
public static void main(String args[])
{
String x = "abc";
String y = x.concat("def").toUpperCase().replace('C', 'x');
System.out.println("y = " + y);

}
}
```

Output

ABxDEF

abc

0	1	2	3	4	5	6	7
a	b	c					

x.concat("def") → abcdef

0	1	2	3	4	5	6	7
a	b	c	d	e	f		

toUpperCase() → ABCDEF

0	1	2	3	4	5	6	7
A	B	C	D	E	F		

replace('C', 'x') → ABxDEF

0	1	2	3	4	5	6	7
A	B	x	D	E	F		

Q1) Can we compare array of String and StringBuffer ?

Eg:

```
String str = "Java";
StringBuffer sbr = new StringBuffer("Java");
if (str.equals(sbr) )
System.out.println("Str and Sbr are equal");
else
System.out.println("Str and Sbr are not equal");
```

Ans: Yes
But the result is incorrect.

Q2) How to get the correct result ?

Eg:

```
String str = "Java";
StringBuffer sbr = new StringBuffer("Java");
if (str.equals(sbr.toString() ) )
System.out.println("Str and Sbr are equal");
else
System.out.println("Str and Sbr are not equal");
```

Ans:
By Using toString() method with StringBuffer

Q3) Can we use == between String and StringBufffer ?

Eg:

```
String str = "Java";
StringBuffer sbr = new StringBuffer("Java");
if (str == sbr )
System.out.println("Str and Sbr are == ");
else
System.out.println("Str and Sbr are not ==");
```

Ans: No,
They are incomparable.

Q4) How to use == between String and StringBuffer

Eg:

```
String str = "Java";
StringBuffer sbr = new StringBuffer("Java");
if (str == sbr.toString() )
System.out.println("Str and Sbr are == ");
else
System.out.println("Str and Sbr are not ==");
```

Ans: Yes
Use to String() method with String Buffer.

P4) WAJP to check whether given string is a palindrome or not.

Program:

```
class PalString {
public static void main(String args[ ]) {
String str = args[0];
int i,j, len = str.length( );
boolean flag = true;
System.out.println("String :"+str);
for( i=0,j=len-1; i<len/2; i++,j--) {
    if(str.charAt(i)==str.charAt(j))
     flag=true;
    else {
       flag=false;
      break;
    }
}
if( flag == true )
System.out.println("String is palindrome");
else
System.out.println("String is not palindrome");
}
}
```

Output:

```
>javac PalString.java
>java PalString MALAYALAM
String :MALAYALAM

String is palindrome
```

P5) WAJP to check whether given string is a palindrome or not using StringBuffer.

Program:

```
class PalStringBuffer {
public static void main(String args[ ]) {
String str = args[0];
StringBuffer sb = new StringBuffer(str);
sb.reverse();
if ( str.equals(sb.toString())
System.out.println("String is palindrome");
else
System.out.println("String is not palindrome");
 }
}
```

Output:

```
>javac PalStringBuffer.java
>java PalStringBuffer MALAYALAM
String is palindrome

>java PalStringBuffer MAAN
String is not palindrome
```

Test Paper

Q1)

```
class TQ5 {
public static void main(String args[ ]) {
int x = 10;
int y = 20;
String myString = "String is: " + x + y + null;
System.out.println(myString);
}
}
```

What will happen when we try to compile and execute the preceding program?

Options:

A. It will produce the output as: String is : 30.
B. It will produce the output as: String is: 1020
C. It will produce the output as: String is: 1020null
D. It will give compilation error.

Concate

Solution: C

Q2)

```
class TQ5a {
public static void main(String args[ ]) {
int x = 10;
int y = 20;
String myString = "String is: " + x - y ; // (x-y) will work
System.out.println(myString);
}
}
```

What will happen when we try to compile and execute the preceding program?

Options:

A. It will produce the output as: String is : 10-20
B. It will produce the output as: String is 1020
C. It will produce the output as: String is: -10
D. It will give compilation error.

Solution: D

Q3) Given:

```
1. public class TestString1 {
2. public static void main(String[] args) {
3. String str = "420";
4. str += 42;
5. System.out.print(str);
6. }
7. }
```

What is the output?

Options:

A. 42
B. 420
C. 462
D. 42042
E. Compilation fails.
F. An exception is thrown at runtime.

Solution: D

Q4)
Which expression will extract the substring "kap", given the following declaration:

String str = "kakapo";

Select the one correct answer.

Options:

(a) str.substring(2, 2)
(b) str.substring(2, 3)
(c) str.substring(2,4)
(d) str.substring(2,5)
(e) str.substring(3, 3)

Solution: d

Q5)
Which statement about the trim() method of the String class is true?
Select the one correct answer.

Options:

(a) It returns a string where the leading white space of the original string has been removed.
(b) It returns a string where the trailing white space of the original string has been removed.
(c) It returns a string where both the leading and trailing white space of the original string has been removed.
(d) It returns a string where all the white space of the original string has been removed.
(e) None of the above.

Solution: C

Q6) Given:

```
11. public static void test(String str) {
12. int check = 4;
13. if (check = str.length()) {
14. System.out.print(str.charAt(check -= 1) +", ");
15. } else {
16. System.out.print(str.charAt(0) + ", ");
17. }
18. }
```

and the invocation:

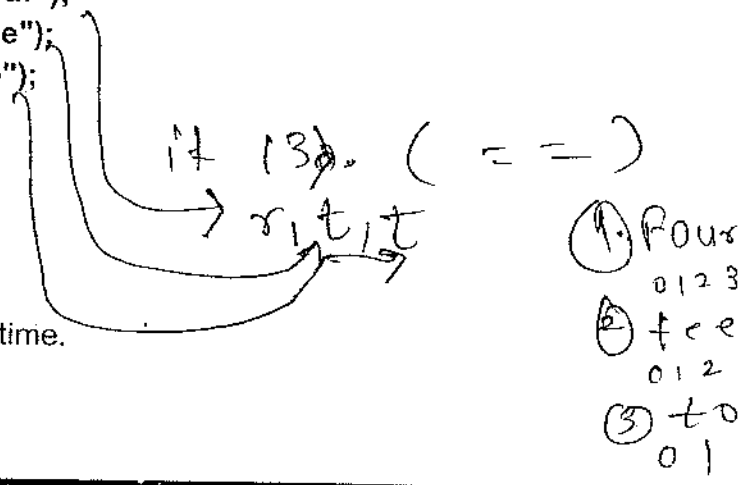
```
21. test("four");
22. test("tee");
23. test("to");
```

What is the result?

Options:

A. r, t, t,
B. r, e, o,
C. Compilation fails.
D. An exception is thrown at runtime.

Solution:



Q7) What will the following program print when run?

```
public class Uppity {
public static void main(String[ ] args) {
String str1 = "lower", str2 = "LOWER", str3 = "UPPER";
str1.toUpperCase();
str1.replace("LOWER","UPPER");
System.out.println((str1.equals(str2)) + " " + (str1.equals(str3)));
}
}
```

Select the one correct answer.

Options:

(a) The program will print false true
(b) The program will print false false
(c) The program will print true false.
(d) The program will print true true.
(e) The program will fail to compile.

Solution: b

Q8) Given:

```
2. public class Maize {
3. public static void main(String[ ] args) {
4. String s = "12";
5. s.concat("ab");
6. s = go(s);
7. System.out.println(s);
8. }
9. static String go(String s) {
10. s.concat("56");
11. return s;
12. } }
```

What is the result?

Options:

A. ab
B. 12
C. ab56
D. 12ab
E. 1256
F. 12ab56
G. Compilation fails.

Solution:

Q9) Given:

```
11. String[ ] elements = { "for", "tea", "too" };
12. String first = (elements.length > 0) elements[0] : null;
```

What is the result?

Options:

A. Compilation fails.
B. An exception is thrown at runtime.
C. The variable first is set to null.
D. The variable first is set to elements[0].

Solution:

Q10) What is the outcome of the following statements? (Select one answer.)

```
6. String s1 = "Canada";
7. String s2 = new String(s1);
8. if(s1 == s2) {
9. System.out.println("s1 == s2");
10. }
11. if(s1.equals(s2)) {
12. System.out.println("s1.equals(s2)");
13. }
```

Options:

A. There is no output.
B. s1 == s2
C. s1.equals(s2)
D. Both B and C

Solution:

Q11) What is the result of the following code?

```
4. String s = "Hello";
5. String t = new String(s);
6.
7. if("Hello".equals(s)) {
8. System.out.print("one");
9. }
10.
11. if(t == s) {
12. System.out.print("two");
13. }          14.
15. if(t.equals(s)) {
16. System.out.print("three");
17. }
```

Options:

A. one
B. onethree
C.twothree
D. onetwothree
E. The code does not compile.

Solution: B

***Q12)** Given:

```
11. public static void main(String[ ] args) {
12. String str = "null";
13. if (str == null) {
14. System.out.println("null");
15. } else if (str.length() == 0) {
16. System.out.println("zero");
17. } else {
18. System.out.println("some");
19. }
20. }
```

Options:

A. null
B. zero
C. some
D. Compilation fails.
E. An exception is thrown at runtime.

Solution: D

Q13) Given:

```
22. public void go() {
23. String o = "";
24. z:
25. for(int x = 0; x < 3; x++) {
26. for(int y = 0; y < 2; y++) {
27. if(x==1) break;
28. if(x==2 && y==1) break z;
29. o = o + x + y;
30. }
31. }
32. System.out.println(o);
33. }
```

What is the result when the go() method is invoked?

Options:

A. 00
B. 0001
C. 000120
D. 00012021
E. Compilation fails.
F. An exception is thrown at runtime.

Solution: C

Q14) Given:

```
3. public class Breaker {
4. static String o = "";
5. public static void main(String[ ] args) {
6. o = o + 2;
7. z:
8. for(int x = 3; x < 8; x++) {
9. if(x==4) break;
10. if(x==6) break z;
11. o = o + x;
12. }
13. System.out.println(o);
14. }
15. }
```

What is the result?

Options:

A. 23
B. 234
C. 235
D. 2345
E. 2357
F. 23457
G. Compilation fails.

Solution: A.

Q15) What is the result of the following code?

```
7. StringBuilder sb = new StringBuilder();
8. sb.append("aaa").insert(1, "bb").insert(4, "ccc");
9. System.out.println(sb);
```

Options:

A. bbaaaccc
B. abbaaccc
C. abbaccca
D. bbaaccca
E. The code does not compile.

Solution:

Q16) Given the following:

```
public class StringIndexMute{
public static void main(String[ ] args){
StringBuilder str = new StringBuilder("0123 456 ");
if (str.length() == 9)
str.insert(9, "abcde");
str.delete(2,5);
System.out.println(str.indexOf("d"));
}}
```

Options:

A. 9
B. 8
C. 7
D. -1
E. Compilation fails.

Solution:

Q17)

```
class TQ1 {
public static void main(String args[ ]) {
System.out.println(new StringBuffer("good day").deleteCharAt(4).substring(3,
5) );
}}
```

What will happen when we try to compile and execute the preceding program?

Options:

A. It will produce the output as "dd"
B. It will produce the output as "d d"
C. It will produce the output as "d day".
D. It will give compile time error.

Solution:

x= 400401

y= 400

Q18) Given:

```
1. public class KungFu {
2. public static void main(String[ ] args) {
3. int x = 400;
4. int y = x;
5. x++;
6. StringBuilder sb1 = new StringBuilder("123");
7. StringBuilder sb2 = sb1;
8. sb1.append("5");
9. System.out.println((x==y) + " " + (sb1==sb2));
10. }
11. }
```

1235

sb1 sb2

1255

What is the result?

Options:

A. true true
B. false true
C. true false
D. false false
E. Compilation fails.

Q19) Which two scenarios are NOT safe to replace a StringBuffer object with a StringBuilder object? (Choose two.)

Options:

A. When using versions of Java technology earlier than 5.0.
B. When sharing a StringBuffer among multiple threads.
C. When using the java.io class StringBufferInputStream.
D. When you plan to reuse the StringBuffer to build more than one string.

Solution: A & B

Q20) Which of the following statements are true about the String, StringBuffer and StringBuilder classes? (Choose all that apply).

Options:

A. StringBuffer's objects are mutable
B. StringBuilder's objects are mutable
C. The methods of StringBuilder class are synchronized.
D. String's objects are mutable.

Solution: A B

Q21) Given :

1. **StringBuilder sb1 = new StringBuilder ("123");**
2. **String s1 = "123";**
3. **// insert code here**
4. **System.out.println(sb1 + " " + s1);**

Which code fragment, inserted at line 3 , outputs "123abc 123abc" ? (Select one)

Options:

A. sb1.append("abc"); s1.append("abc");
B. sb1.append("abc"); s1.concat("abc");
C. sb1.concat("abc"); s1.append("abc");
D. sb1.concat("abc"); s1.concat("abc");
E. sb1.append("abc"); s1 = s1.concat("abc");
F. ~~sb1.append("abc"); s1 = s1.concat("abc");~~
G. sb1.append("abc"); s1 = s1 + s1.concat("abc");

Solution:

Q22) Given:

1. **public class TestString3 {**
2. **public static void main(String[ ] args) {**
3. **// insert code here**
5. **System.out.println(s);**
6. **}**
7. **}**

Which two code fragments, inserted independently at line 3, generate the output **4247**? (Choose two.)

Options:

A. String s = "123456789"; s = (s-"123").replace(1,3,"24") - "89";	**B.** StringBuffer s = new StringBuffer("123456789"); s.delete(0,3).replace(1,3,"24").delete(4,6);
C. StringBuffer s = new StringBuffer("123456789"); s.substring(3,6).delete(1,3).insert(1, "24");	**D.** StringBuilder s = new StringBuilder("123456789"); s.substring(3,6).delete(1,2).insert(1, "24");
E. StringBuilder s = new StringBuilder("123456789"); s.delete(0,3).delete(1,3).delete(2,5).insert(1, "24");	

Solution:

(B) 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8

4 5 6 7 8 9
0 1 2 3 4 5

4 2 4 7 8 9
0 1 2 3 4 5

4247

(C) 1 2 5 4 5 1 7 8 9
0 1 2 3 4 5 6 7 8

4 5 6
0 1 2

4
0

~~4 2 4~~ ✗

(D) 1 2 3 9 5 6 7 8 9
0 1 2 3 ———

4 5 6
0 1 2

4 6
0 1

4246 ✗

(2) 1 2 ———
0 ———

4 5 6 7 8 9
0 1 2 3 4 5

4 7 8 9 —
0 1 2 3 4

4 7
0 1

4247