Q1.

```
class TestString{
	public static void main(String args[]){
		String s1="abc";
		String s2=new String("abc");
		String s3=new String("abc");
		String s4="abc";
		System.out.println("s1==s2 :"+(s1==s2));
		System.out.println("s1==s3 :"+(s1==s3));
		System.out.println("s1==s4 :"+(s1==s4));
		System.out.println("s2==s3 :"+(s2==s3));
	}
}
```

Q2.

```
class TestString{
	public static void main(String args[]){
		String s1="abc";
		s1.concat("def");
		System.out.println("s1 :"+s1);
	}
}
```

Q3.

```
	class TestString{
		public static void main(String args[]){
			String s1="abc";
			String s2=s1;
			s1=s1.concat("def");
			System.out.println("s1 :"+s1);
			System.out.println("s2 :"+s2);
		}
	}
```

Q4.

```
	class TestString{
		public static void main(String args[]){
			String s1="abc";
			String s2=s1;
			s1=s1+"abc";
			System.out.println("s1 :"+s1);
			System.out.println("s2 :"+s2);
		}
	}
```

Q5.

```
	class TestString{
		static String m(){
			String name="abc";
			return name;
		}
		public static void main(String args[]){
			String s1="abc";
			String s2=m();
			System.out.println("s1==s2 :"+(s1==s2));
		}
	}
```

Q6.

```
class TestString{
	static String m(){
		String name=new String("abc");
		return name;
	}
	public static void main(String args[]){
		String s1="abc";
		String s2=m();
		System.out.println("s1==s2 :"+(s1==s2));
	}
}
```

Q7.

```
class TestString{
	public static void main(String args[]){
		String s1="abc";
		System.out.println(s1=="abc");
	}
}
```

Q8.

```
class TestString{
	public static void main(String args[]){
		String s1="abc";
		String s2="ab";
		String s3=s2+"c";
		System.out.println(s1==s3);
	}
}
```

Q9.

```
class TestString{
	public static void main(String args[]){
		String s1="abc";
		final String s2="ab";
		String s3=s2+"c";
		System.out.println(s1==s3);
	}
}
```

Q10. **String Constructors**

```
class TestString{
	public static void main(String args[]){
		String s1=new String(); //No parameter
		String s2=new String("abc");//String parameter
		char ch[]={'a','b','c'};
		String s3=new String(ch);//char array
		byte br[]={97,98,99};
		String s4=new String(br);//byte array
		System.out.println(s1+" "+s2+" "+s3+" "+s4);
	}
}
```

Q11. Special Constructor methods

char	charAt(int index)	Returns the character at the specified index

```
class TestCharAt{
	public static void main(String args[]){
		String s1=new String("abcXdef");
		char ch1=s1.charAt(0);
		char ch2=s1.charAt(3);
		System.out.println(ch1+" "+ch2);

		char ch3=s1.charAt(-1); //Run time error StringIndexOutOfBoundsException
		char ch4=s1.charAt(7);  //Run time error StringIndexOutOfBoundsException
	}
}
```

Q12.

String	**concat**(String str)	Concatenates the specified string to the end of this string.

```
class Testconcat{
	public static void main(String args[]){
		String s1=new String("abc");
		s1.concat("xyz");
		System.out.println("Now s1 :"+s1);
		s1=s1.concat("xyz");
		System.out.println("Now s1 :"+s1);
	}
}
```

Q13.

boolean	**equals**(Object anObject)	Compares this string to the specified object.

```
class Test_equals{
	public static void main(String args[]){
		String s1=new String("abc");
		String s2=new String("abc");
		String s3=new String("def");
		System.out.println("s1 & s2 :"+s1.equals(s2));
		System.out.println("s1 & s3 :"+s1.equals(s3));
		System.out.println("s1==s2 :"+(s1==s2));
		System.out.println("s1==s3 :"+(s1==s3));
	}
}
```

Q14.

boolean **equalsIgnoreCase**(String anotherString)	Compares this `String` to another `String`, ignoring case considerations.

```
class Test_equalsIgnoreCase{
	public static void main(String args[]){
		String s1=new String("abc");
		String s2=new String("ABc");
		String s3=new String("abc");
		System.out.println("s1 & s2 :"+s1.equalsIgnoreCase(s2));
		System.out.println("s1 & s3 :"+s1.equalsIgnoreCase(s3));
	}
}
```

Q15.

`int length()`	Returns the length of this string

```
	class TestLength{
		public static void main(String args[]){
		String s1=new String("abcd abc");
		System.out.println("No of characters :"+s1.length());
	}
}
```

Q16.

`String substring(int beginIndex)`	Returns a new string that is a substring of this string.

```
class Test{
	public static void main(String args[]){
		String s1="Sun Certified Java Programmer";
		String s2=s1.substring(14);
		System.out.println(s2);
	}
}
```

Q17.

`String substring(int beginIndex,int endIndex)`	Returns a new string that is a substring of this string.

```
class Test{
	public static void main(String args[]){
		String s1="Sun Certified Java Programmer";
		String s2=s1.substring(0,3);
		System.out.println("0-3 :"+s2);
		String s3=s1.substring(4,14);
		System.out.println("4-14 :"+s3);
		// String s4=s1.substring(5,3); //wrong index
	}
}
```

Q18.

`String toLowerCase()`	Converts all of the characters in this String to lower case using the rules of the default locale.
`String toUpperCase()`	Converts all of the characters in this `String` to upper case using the rules of the default locale.

```
class Test{
	public static void main(String args[]){
		String s1="EnGliSh";
		System.out.println("Normal Case :"+s1);
		System.out.println("Lower Case :"+s1.toLowerCase());
		System.out.println("Upper Case :"+s1.toUpperCase());
	}
}
```

Q19.

```
class Test{
	public static void main(String args[]){
		String s1="ijts";
		String s2=s1.toUpperCase();
		String s3=s1.toLowerCase();
		System.out.println("s1==s2 :"+(s1==s2));
		System.out.println("s1==s3 :"+(s1==s3));
	}
}
```

Q20.

String **trim**()	Returns a copy of the string, with leading and trailing whitespace omitted.

```
class Test{
	public static void main(String args[]){
		String s1=" EnGliSh ";
		System.out.println("No of character :"+s1.length());
		s1.trim();
		System.out.println("No of character :"+s1.length());

		String s2=s1.trim();
		System.out.println("No of character :"+s2.length());
	}
}
```

StringBuffer

Q21. StringBuffer

```
class TestStringBuffer{
	public static void main(String args[]){
		StringBuffer sb1=new StringBuffer("SCJP @ ");
		StringBuffer sb2=sb1;
		sb1.append("IJTS"); //appends to the end of the String
		System.out.println(sb1);
		System.out.println(sb2);
	}
}
```

Q22. StringBuffer Special Methods

StringBuffer **append**(boolean b)	Appends the string representation of the boolean argument to the string buffer.
StringBuffer **append**(char c)	Appends the string representation of the char argument to this string buffer.
StringBuffer **append**(double d)	Appends the string representation of the double argument to this string buffer.
StringBuffer **append**(float f)	Appends the string representation of the float argument to this string buffer.
StringBuffer **append**(int i)	Appends the string representation of the int argument to this string buffer.
StringBuffer **append**(long l)	Appends the string representation of the long argument to this string buffer.
StringBuffer **append**(Object obj)	Appends the string representation of the Object argument to this string buffer.
StringBuffer **append**(String str)	Appends the string to this string buffer.
StringBuffer **append**(StringBuffer sb)	Appends the specified StringBuffer to this StringBuffer.

```
class A{
	public String toString(){return "Class A";}
}
class TestStringBuffer{
	public static void main(String args[]){
		StringBuffer sb1=new StringBuffer();
		sb1.append(10);
		System.out.println(sb1);
		sb1.append(10.99);
		System.out.println(sb1);
		sb1.append("VV");
		System.out.println(sb1);
		sb1.append(14.5f);
		System.out.println(sb1);
		sb1.append(new A());
		System.out.println(sb1);
	}
}
```

Q23. insert()

`StringBuffer` **`insert`**`(int offset, boolean b)`	Inserts the string representation of the boolean argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, char c)`	Inserts the string representation of the char argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, double d)`	Inserts the string representation of the double argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, float f)`	Inserts the string representation of the float argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, int i)`	Inserts the string representation of the second int argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, long l)`	Inserts the string representation of the long argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, Object obj)`	Inserts the string representation of the Object argument into this string buffer.
`StringBuffer` **`insert`**`(int offset, String str)`	Inserts the string into this string buffer.

```
class TestStringBuffer{
        public static void main(String args[]){
                StringBuffer sb1=new StringBuffer("abcd");
                sb1.insert(2,false);
                System.out.println(sb1);
                sb1.insert(6,new A());
                System.out.println(sb1);
                }
}
```

Q24.   .

`StringBuffer` **`delete`**`(int start, int end)`	Removes the characters in a substring of this `StringBuffer`.

```
class TestStringBuffer{
        public static void main(String args[]){
                StringBuffer sb1=new StringBuffer("abcxyzdef");
                sb1.delete(3,6);
                System.out.println("abcxyzdef deleted 3-5 :"+sb1);
        }
}
```

Q25.

`StringBuffer` **`reverse`**`()`	Causes this character sequence to be replaced by the reverse of the sequence.

```
class TestStringBuffer{
        public static void main(String args[]){
                StringBuffer sb1=new StringBuffer("STJI @ PJCS");
                sb1.reverse();
                System.out.println("Reverse :"+sb1);
        }
}
```

Q26.

`String` **`toString`**`()`	Returns a string representing the data in this sequence.

```
class TestStringBuffer{
        public static void main(String args[]){
                StringBuffer nicNo=new StringBuffer("856573702V");
                //String nic=nicNo;  //Compile Error
                String nic=nicNo.toString();
                System.out.println("NIC numer :"+nic);
        }
}
```