String Handling

Supported by :

java.lang.String

java.lang.StringBuffer

java.lang.StringBuilder

String

- Strings are immutable objects of type java.lang.String.
- Meaning of Immutablility : We can't change the sequence of Characters encapsulated in a String Object. Each operation on that object will produce a new String object.

- Ways of Creating String Objects : 2

1.Using new Keyword

»**String s1 = new String();**

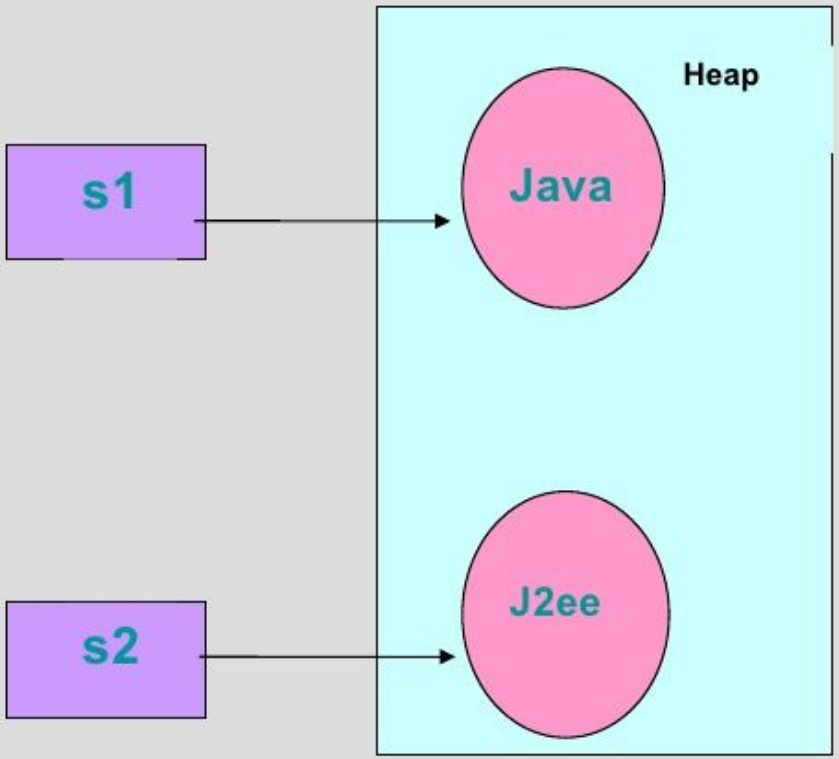
»**String s2 = new String("Java");**

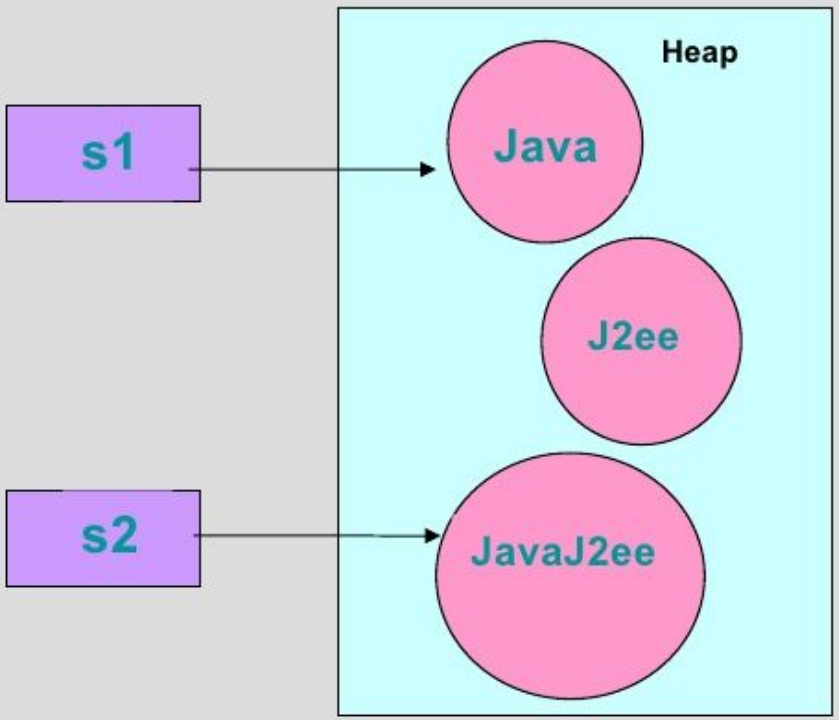
2.Using String literal

»**String s3 = "J2EE"**

- Understanding the immutability :
  - String s1 = new String("Java");
  - String s2 = new String("J2ee");

s2=s1.concat(s2);



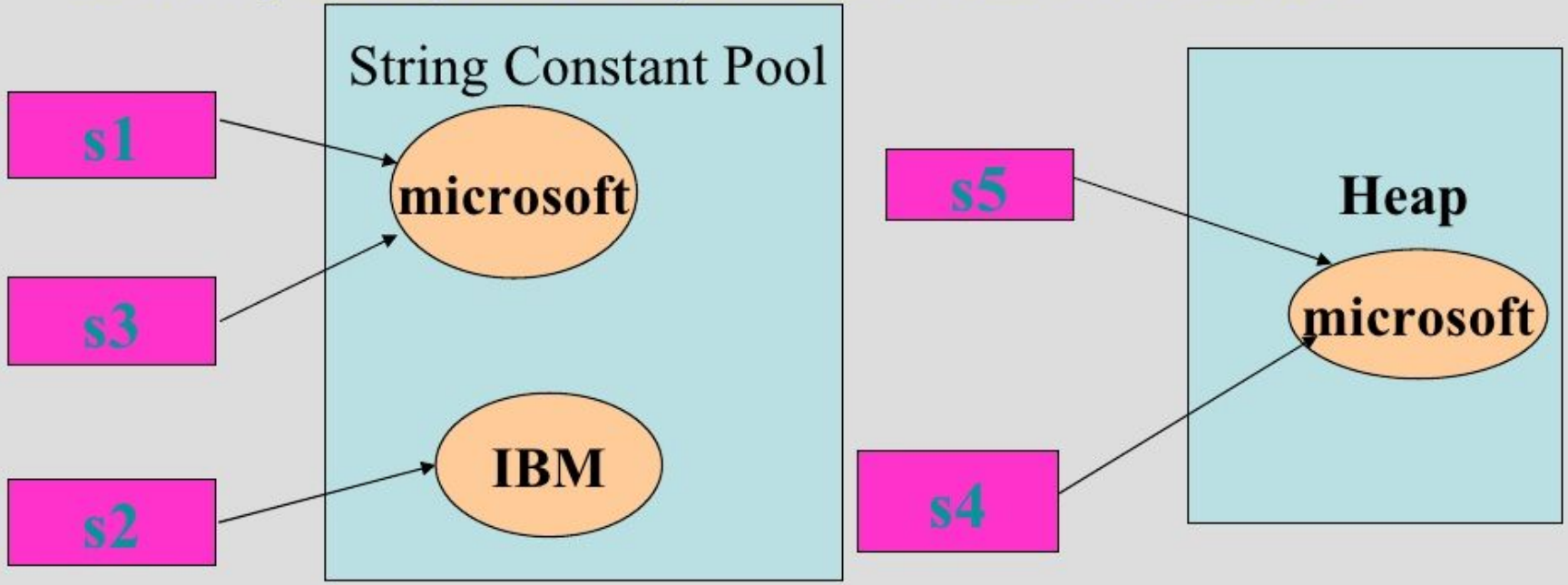


- Which one was immutable? String reference variable or String Object itself.
- Count the total no. of objects created. How many String objects can be referred to by our application in the last example? How many are missing?

String Literal & String Constant Pool

- String s1 = “microsoft”;
- String s2 = “IBM”;
- String s3 = “microsoft”;
- String s4 = new String(“microsoft:”);
- String s5=s4;

How may String literal objects have been constructed?



String Literal & String Constant Pool

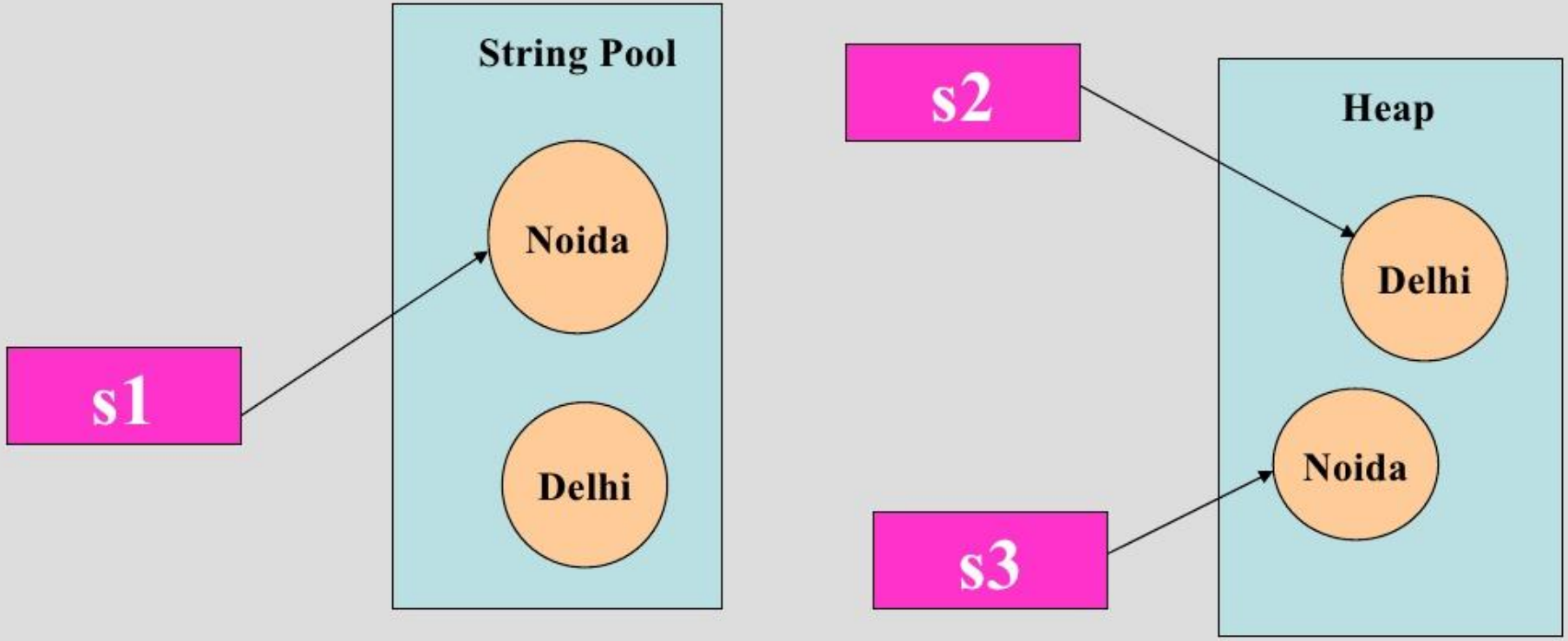
- It a special memory area set aside by JVM to make Java more memory efficient.

- **When a String literal is encountered, the pool is checked to see if an identical String already exists. If it's there, no new String literal object is created, we'll simply have an additional reference to the same existing object.**

- **Now, if several reference variables refer to the same String object without even knowing it. It would be very bad if anyone could change the String's value. So Strings have been made immutable and java.lang.String has been declared final.**

Understanding the different ways to create String Objects

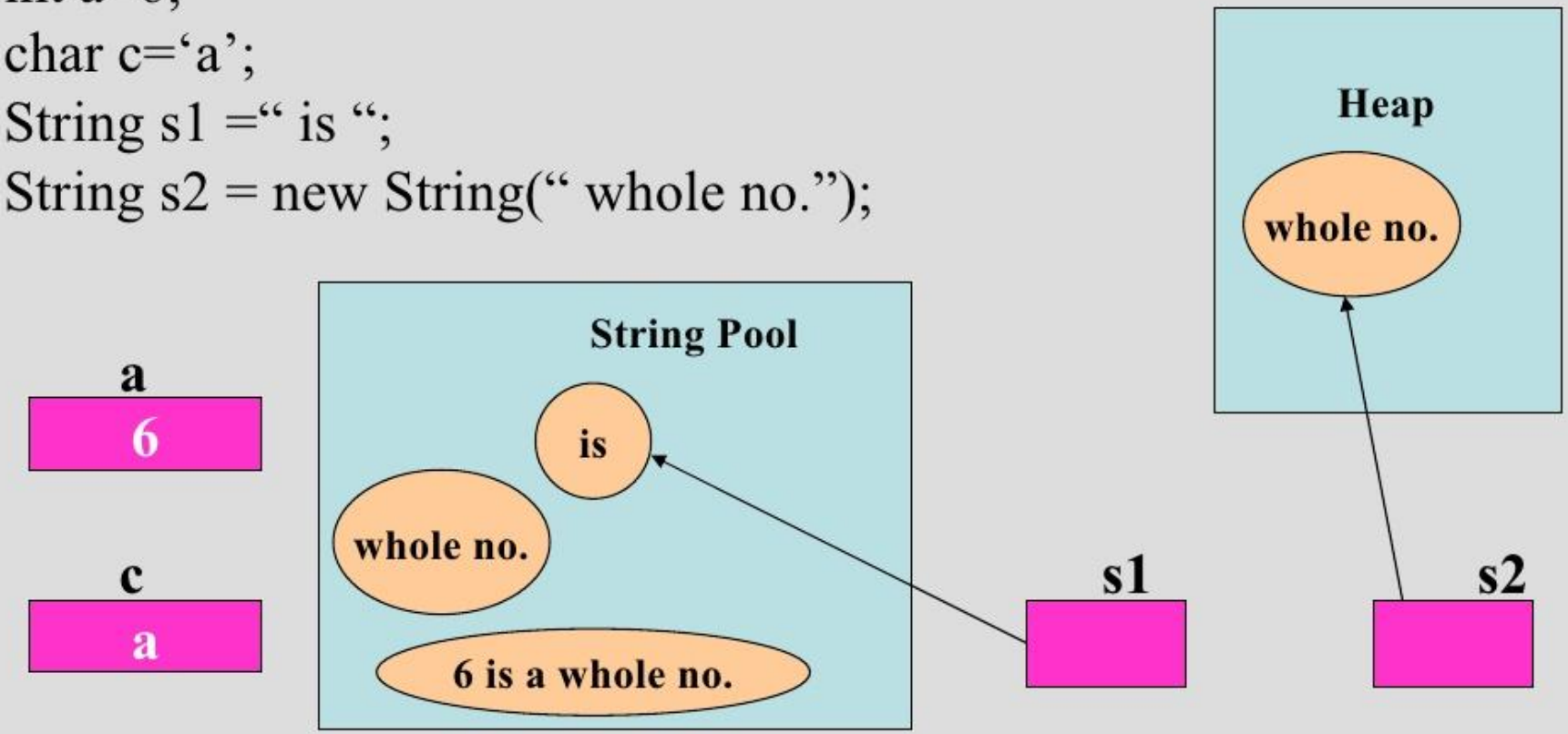
- String s1 = “Noida”;
- String s2 = new String(“Delhi”);
- String s3 = new String(“Noida”);



- Note that String literal “Delhi” has been placed in the pool.

+ & += have been overloaded to work on Strings.

```
int a=6;
char c='a';
String s1 =" is ";
String s2 = new String(" whole no.");
```




```
System.out.println(a+s1+c+s2);
```

Working with Strings : Using public methods defined in the java.lang.String

public char charAt(int index)

returns the character with the index specified.

public String concat(String s)

concatenates s at the end of the string invoked & returns the reference of new String Object.

public Boolean equalsIgnoreCase(String s)

Compares 2 String Objects irrespective of their character case.

public int length()

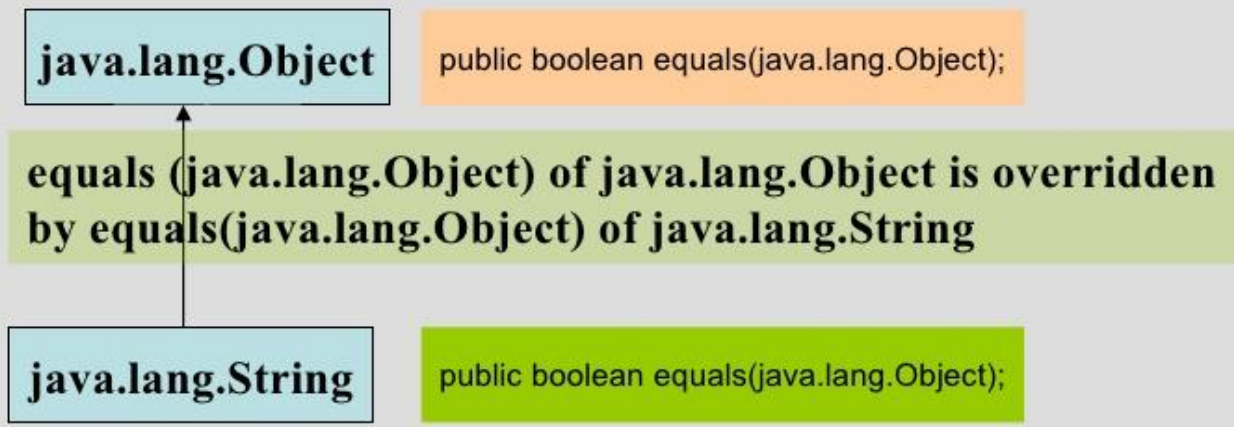
returns the no. of characters of the character sequence encapsulated in the string object.

length() of String class is different than the length that we use to get no. of elements in an array.

```
String s="winter";
char c[]=new char[]{'s','u','m','m','e','r'};
s.length;   //invalid
c.length(); //invalid
```

Shallow & Deep Comparison

```
String s1 = "star";  String s2 = s1 ;
String s3 = new String("star");
System.out.println( s1== s2 );
System.out.println( s1.equals(s2) );
System.out.println( s1 == s3 );
System.out.println( s2.equals(s3) );
```



Output :

true

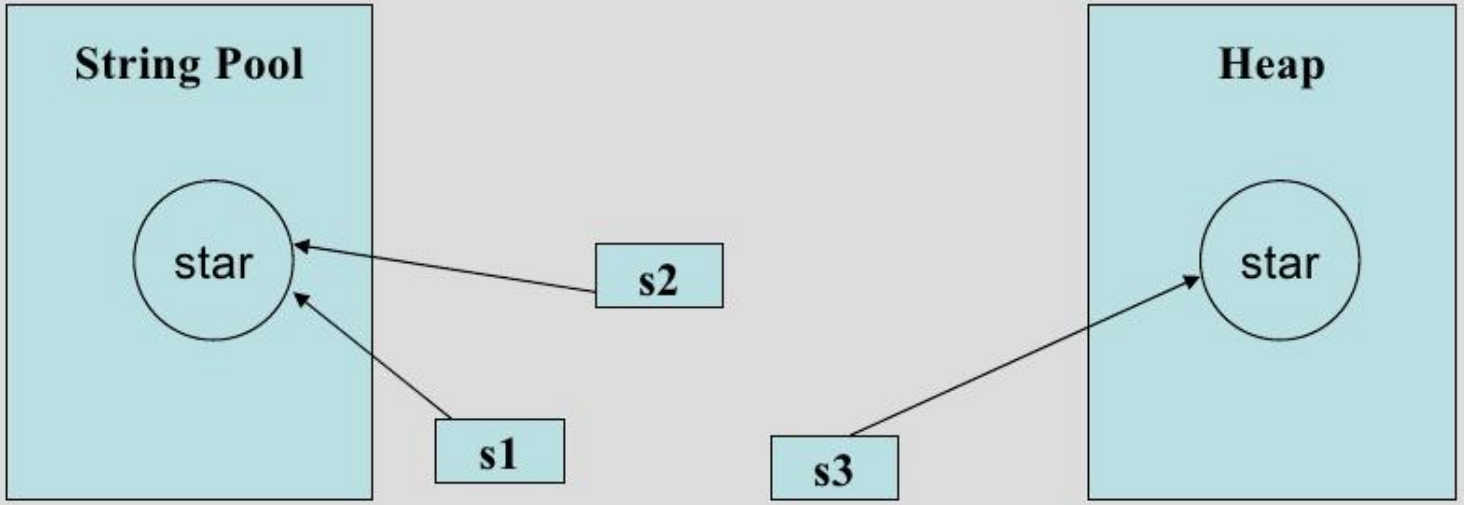
true

false

true

Shallow & Deep Comparison contd.

- == does shallow comparison,means calls equals() from Object class that compares the refrences not their character sequences.
- While equals() of String class doesn't compares their refrences. Instead it compares their character sequences.



StringBuffer

- Objects of StringBuffer type are used when we have to do a lot of modification to strings of characters.

- StringBuffer objects are mutable.

```
String s = “keyboard”;
s.concat(“mouse”);
System.out.println(s);    // output ?

String s = “Tom”
s=s.concat(“cat”);
System.out.println(s);   // output ?
```

StringBuffer contd.

- All of the StringBuffer methods operate on the content of the object invoked on.The methods won't produce intermediate objects.

```
StringBuffer sb = "micro";
sb.append("soft");
System.out.println(sb);    // output ?
```

Important methods of StringBuffer

public synchronized StringBuffer append(String s)
public synchronized java.lang.StringBuffer append(char[]);
public synchronized java.lang.StringBuffer append(char[], int, int);
public synchronized java.lang.StringBuffer append(boolean);
public synchronized java.lang.StringBuffer append(char);
public synchronized java.lang.StringBuffer append(int);
public synchronized java.lang.StringBuffer append(long);
public synchronized java.lang.StringBuffer append(float);
public synchronized java.lang.StringBuffer append(double);
public synchronized java.lang.StringBuffer append(java.lang.Object);
public synchronized java.lang.StringBuffer append(java.lang.StringBuffer);
public java.lang.StringBuffer append(java.lang.CharSequence);

StringBuffer sb = new StringBuffer();
sb.append("5>6=");
sb.append(5>6);

Important methods of StringBuffer

public synchronized StringBuffer insert(int offset, String s)
public java.lang.StringBuffer insert(int, boolean);
public synchronized java.lang.StringBuffer insert(int, char);
public java.lang.StringBuffer insert(int, int);
public java.lang.StringBuffer insert(int, long);
public java.lang.StringBuffer insert(int, float);
public java.lang.StringBuffer insert(int, double);

```
StringBuffer sb = new StringBuffer(“abcdefg");
sb.insert(2, “   ");
System.out.println( sb );  // output ab  cdefg
```

public synchronized StringBuffer reverse()

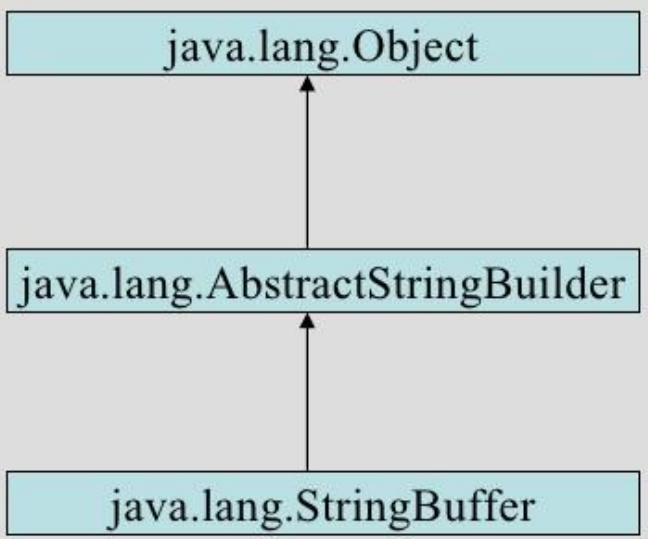
```
StringBuffer sb = new StringBuffer(“pooL”);
sb.reverse();
System.out.println(sb);   // output Loop
```

public String toString()

StringBuffer contd.

```
StringBuffer sb1 = new StringBuffer("microsoft");
StringBuffer sb2 = new StringBuffer("microsoft");

System.out.println( sb1 == sb2 );
System.out.println( sb1.equals(sb2) );
```



public boolean equals(java.lang.Object);

Output :

false

false