

Java : Strings

Salman Khan

ID: 1309004

Session: 2013-14

Dept. of ICT

Comilla University

Outline



- What

is string ?

- How to create String object?
- Some important operations of strings
- Some important string class methods
- StringBuffer class & StringBuilder class
- Difference between String and StringBuffer
- Difference between StringBuffer and StringBuilder

What is String ?



- Strings, which are widely used in Java programming, are *a sequence of characters*. In

Java programming language, strings are treated as objects.

- The Java platform provides the **String class** to create and manipulate strings.

Important Notes about String

- String
— Object
or

Primitive?

- Strings could be considered a primitive type in Java, but in fact they are not. As a String is actually made up of an array of char primitives.

- String objects are **immutable!**

- That means once a string object is created it cannot be altered. For mutable string, you can use *StringBuffer* and *StringBuilder* classes.

[An object whose state cannot be changed after it is created is known as an Immutable object. String, Integer, Byte, Short, Float, Double and all other wrapper class's objects are immutable.]

How to create String object?



There are two ways to create String

object: 1.By string literal

For Example:


```
String s="welcome";
```

2.By *new* keyword

For Example:

```
String s=new String("Welcome");
```

Java String Example



```
public class StringExample {
```

```
    public static void main(String args[]){
```

```
        String s1="java";
```

```
        //creating string by java string literal
```


Output:

java
strings
example

Some important operations of strings



String Concatenation

- String Comparison
- Substring
- Length of String etc.

String Concatenation



- There are 2 methods to concatenate two or more string.

```
public class StringExample {  
  
    public static void main(String args[]){  
  
        String s = "Hello";  
        String str = " World";  
        //Using concat() method  
        String str1 = s.concat(str);  
        String str2 = "Hello".concat(" World");  
        //Using + operator  
        String str3 = s + str;  
        String str4 = "Hello"+" World";  
  
        System.out.println(str1);  
        System.out.println(str2);  
        System.out.println(str3);  
        System.out.println(str4);  
    }  
}
```

General Output

```
-----  
Hello World  
Hello World  
Hello World  
Hello World
```

```
Process completed.
```

String Comparison



String Comparison Example

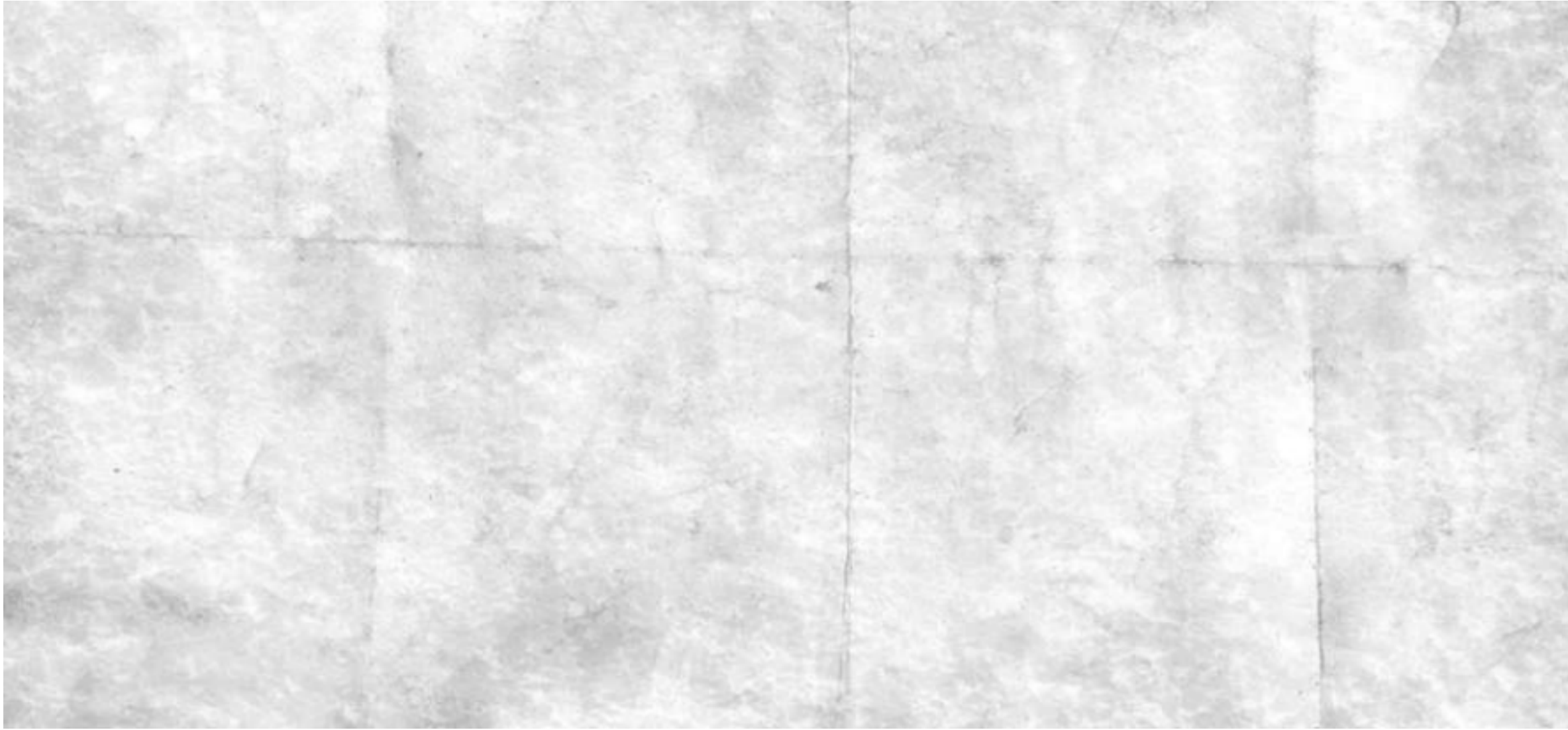
General Output

```
-----  
true  
false  
true  
false  
-1  
0  
1  
  
Process completed.
```

```
public static void main(String args[]){

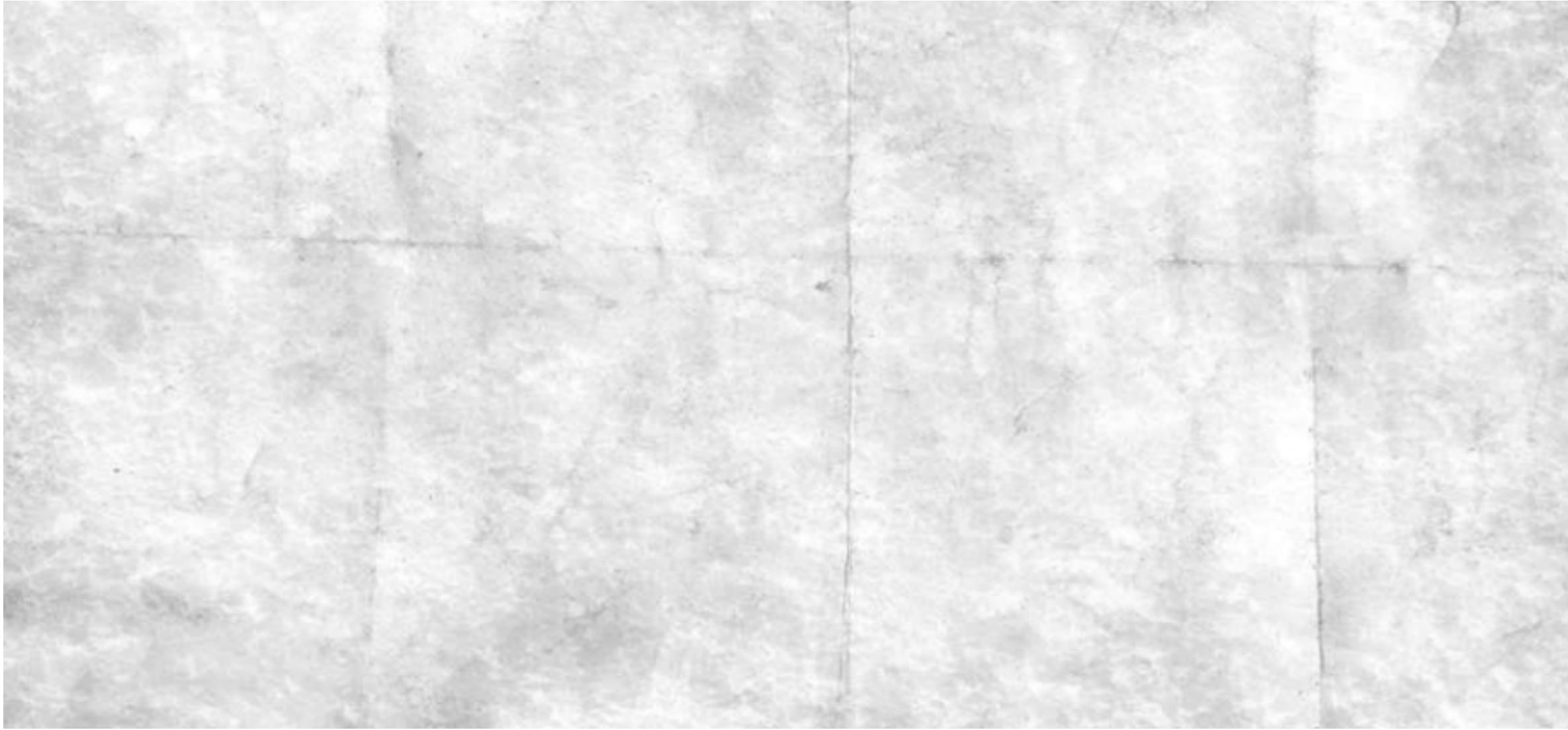
    String s = "Hell";
    String s1 = "Hello";
    String s2 = "Hello";
    String s3 = "Java";
    //Using equals() method
    System.out.println(s1.equals(s2));    //true
    System.out.println(s.equals(s1));    //false
    //Using == operator
    String s4 = new String ("Java");
    System.out.println(s1==s2); //true (because both refer to same instance)
    System.out.println(s3==s4); //false(because s3 refers to instance created in nonpool)
    //By compareTo() method
    System.out.println(s.compareTo(s2));    //return -1 because s1 < s2
    System.out.println(s1.compareTo(s2));    //return 0 because s1 == s3
    System.out.println(s2.compareTo(s));    //return 1 because s2 > s1
}

}
```

Substring in Java





- A part of string is called substring. In other words, substring is a subset of another string.

- We can get substring from the given string object by one of the two methods: 1.

`public String substring(int startIndex):`

2. `public String substring(int startIndex, int endIndex):`

Length of String







- The java string **length()** method finds the length of the string. It returns count of total number of characters.

Some more string class methods

- The



java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on strings.

- Here we will learn about the following methods :

1. charAt()
2. contains()
3. getChars()
4. indexOf()
5. replace()
6. toCharArray()
7. toLowerCase()
8. toUpperCase()

charAt() & contains()







□ charAt() = returns a char value at the given index number. The index number starts from 0. □ contains() = searches the sequence of characters in this string. It returns true if sequence of char values are found in this string otherwise returns false.

getChars()







□ getChars() = copies the content of this string into specified char array. There are 4 arguments passed in getChars() method. It throws

StringIndexOutOfBoundsException if *beginIndex* is greater than *endIndex*.

indexOf()







□ indexOf() = returns index
of given character value
or substring. If it is not

found, it returns -1. The index counter starts from zero.

replace()







□ `replace()` = returns a string replacing all the old char or CharSequence to new char or CharSequence.

toCharArray()







□ `toCharArray()` = converts this string into character array.

toLowerCase() & toUpperCase()







□ `toLowerCase()` = returns the string in lowercase letter. In other words, it converts all characters of the string into lower case letter.

□ toUpperCase() = returns the string in uppercase letter. In other words, it converts all characters of the string into upper case letter.

Java StringBuffer class

■
Java



StringBuffer class is used to create mutable (modifiable) string.

- Important Constructors of **StringBuffer** class:
 - **StringBuffer()**: creates an empty string buffer with the initial capacity of 16.
 - **StringBuffer(String str)**: creates a string buffer with the specified string.
 - **StringBuffer(int capacity)**: creates an empty string buffer with the specified capacity as length.
- Important methods of **StringBuffer** class:
 - **append(String s)**: is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc. □
 - insert(int offset, String s)**: is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
 - **replace(int startIndex, int endIndex, String str)**: is used to replace the string from specified startIndex and endIndex.
 - **delete(int startIndex, int endIndex)**: is used to delete the string from specified startIndex and endIndex.
 - **reverse()**: is used to reverse the string.
 - **capacity()**: is used to return the current capacity.

Java **StringBuilder** class



Java StringBuilder class is used to create mutable (modifiable) string. The Java StringBuilder class is same as StringBuffer class

except that it is non-synchronized.

Example of using StringBuffer & StringBuilder class







Difference between String and StringBuffer

No.	String	StringBuffer
1)	String class is immutable.	StringBuffer class is mutable.
2)	String is slow and consumes more memory when you concat too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when you concat strings.
3)	String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.	StringBuffer class doesn't override the equals() method of Object class.

Difference between StringBuffer and StringBuilder

No.	StringBuffer	StringBuilder
1)	StringBuffer is <i>synchronized</i> i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is <i>non-synchronized</i> i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is <i>less efficient</i> than StringBuilder.	StringBuilder is <i>more efficient</i> than StringBuffer.

Thanks to all