Web Technology 12

String Class & StringBuffer Class

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

String

String

- Once a String Object has been created, you cannot change the characters that comprise that string, i.e. ***String is Immutable***
- To solve this, Java provides a companion class to String called StringBuffer
- StringBuffer objects can be modified after they are created

String Constructor

String Constructor

- **Creating an empty String**:
  - ***String s = new String();***
- **Creating a string that has initial values**:
  - ***String(char chars[])***
  - *char chars[] = {'a', 'b', 'c'};*
    String s = new String(chars);
- **Creating a string as a subrange of a character array**:
  - ***String(char chars[], int startindex, int numchars)***
  - *char chars[] = {'a', 'b', 'c', 'd', 'e', 'f'};*
    String s = new String(chars,2,3);

String Constructor...

String Constructor...

- **Constructing a String object that contains the same character sequence as another String object:**
  - ***String(String obj)***
  - *char c[] = {'J', 'a', 'v', 'a'};*
    String s1 = new String(c);
    String s2 = new String(s1);

String Operations

String Operations

- **String Literals**:
  - For each String literal, Java automatically constructs a String object

    char chars[] = {'a','b','c'};
    String s1 = new String(chars);

  - Using String literals

    String s2 = äbc";

- **String Concatenation**:
  - *String age = "19";*
    String s = "He is "+ age + "years old.";

String Functions

String Functions

String s = "Welcome to demo program";

- **length()**:
  - *int len = s.length();*
- **charAt(n)**:
  - *String fruit = "banana";*
    char ch = fruit.charAt(1);
    System.out.println(ch);
- **getChars(n1, n2, s, n3)**:
  - *int start = 4, end = 8, destoffset;*
    char buf[] = new char[end - start];
    s.getChars(start, end, buf, destoffset);

String Functions...

String Functions...

- **equals(s)**:
  - Returns true if the strings contain the same characters; otherwise false

```
String name1 = "GOOD";
    String name2 = "GooD";
    if (name1.equals (name2)) {
            System.out.println("The names are the same.");
            }
```

- **equalsIgnoreCase(s)**:
  - Similar to equals() by ignoring the cases

```
if (name1.equalsIgnoreCase (name2)) {
            System.out.println("The names are the same.");
            }
```

String Functions...

String Functions...

- **compareTo(s)**:
  - Returns the difference between the first characters in the strings that differ

```
int flag = name1.compareTo (name2);
   if (flag == 0) {
                System.out.println("The names are the same.");
                }
```

- **indexOf(c)**:
  - searches the first occurrence of a character
  - *String fruit = "banana";*
    int index = fruit.indexOf('a');

- **lastIndexOf(c)**:
  - searches the last occurrence of a character
  - *int index = fruit.lastIndexOf('a');*

String Functions...

String Functions...

- **indexOf(c, n)**:
  - used to specify a starting point for the search
  - *String fruit = "banana";*
    int index = fruit.indexOf('a', 2);
- **lastIndexOf(c, n)**:
  - used to specify a ending point for the search
  - *int index = fruit.lastIndexOf('a',2);*
- **substring()**:
  - Extracts a sub string
  - *String org = "Welcome to Java";*
    String result = null;
    result = org.substring(2, 6);
- **concat()**:
  - *String s1 =Ram";*
    String s2 = s1.concat("Hari");
    or
    String s2 = s1 + "Hari";

String Functions...

String Functions...

- **replace()**:
  - Replaces all occurrences of one character in the invoking string with another character
  - *String s = "Hello".replace('l','w');*
- **trim()**:
  - Returns a copy of the involving string from which any leading and trailing whitespace has been removed
  - *String s = "Hello world ".trim();*
- **toUpperCase()**:
  - *String s = "Welcome to test.";*
    String upper = s.toUpperCase();
- **toLowerCase()**:
  - *String lower = s.toLowerCase();*

Non-alphabetical characters, such as digits are unaffected

String Conversion and toString()

String Conversion and toString()

- If you want to represent any object as a string, *toString()* is used. This method returns the string representation of the object
- If you print any object, java compiler internally invokes toString() method on the object. Overriding toString() method returns the desired output
- By overriding the toString() method of the Object class, we can return values of the object, so we don't need to write much code Every class implements toString() because it is defined by **Object**
- *public String toString(){*
  return "Dimensions: "+ height + ", "+ width;
  }

Data Conversion using valueOf()

Data Conversion using valueOf()

- It converts different types of values into string. It is a static method that is overloaded within String for all built-in types
- *public static String valueOf(boolean b)*
  public static String valueOf(char c)
  public static String valueOf(int i)
  public static String valueOf(double d)
- *int data=30;*
  String str=String.valueOf(data);
  System.out.println(str+40);

StringBuffer

- StringBuffer is a peer class of String that provides much of the functionality of Strings
- String is immutable. StringBuffer is mutable. It represents growable and writable character sequence
- StringBuffer may have characters and substring inserted in the middle or appended to the end
- It will automatically grow to make room for such additions

StringBuffer Constructor

StringBuffer Constructor

- **Creating an empty StringBuffer**
  - **StringBuffer sb = new StringBuffer();**
- **Creating size-defined StringBuffer**
  - **StringBuffer sb = new StringBuffer(int size);**
    StringBuffer sb = new StringBuffer(50);
- **Creating String object - based StringBuffer**
  - **StringBuffer sb = new StringBuffer(String str);**
    StringBuffer sb = new StringBuffer("Hello");

StringBuffer Functions

- **length()**
  - finds the current length
  - *StringBuffer sb=new StringBuffer("Hello");*
    int len=sb.length();
- **capacity()**
  - finds the total allocated capacity
  - *StringBuffer sb=new StringBuffer("Hello");*
    int cap=sb.capacity();
- **ensureCapacity()**
  - sets the size of the buffer after a StringBuffer has been constructed
  - *void ensureCapacity(int capacity)*
  - capacity specifies the minimum size of the buffer

StringBuffer Functions...

StringBuffer Functions...

- **setLength()**
  - sets the length of the buffer within a StringBuffer object
  - *void setLength(int len)*
  - len specifies the length of the buffer
  - *sb.setLength(4);*

- **charAt()**
  - obtains the value of a single character
  - char charAt(int pos)
  - pos specifies the index of the character being obtained
  - *StringBuffer sb=new StringBuffer("Hello");*
    System.out.println(Character: "+sb.charAt(2));

- **setCharAt()**
  - sets the value of a character
  - void setCharAt(int pos, char ch)
  - pos specifies the index of the character being obtained
  - *StringBuffer sb=new StringBuffer("Hello");*
    sb.setCharAt(2,'i');

StringBuffer Functions...

StringBuffer Functions...

- **getChars()**
  - copies a substring of a StringBuffer into an array
  - *void getChars (int begin, int end, char[] target, int targetbeg)*

- **append()**
  - concatenates the string representation of any type of data to the end of the invoking StringBuffer object
  - *StringBuffer append(Object ob)*
    StringBuffer append(String str)
    StringBuffer append(int num)
  - String.valueOf() is called for each parameter to obtain its string representation
  - *String s=null;*
    int a=100;
    StringBuffer sb=new StringBuffer (40);
    s=sb.append(ä=").append(a).append("!").toString();

StringBuffer Functions...

- **insert()**
  - inserts one string into another
  - *StringBuffer insert(int index, String str)*
    StringBuffer insert(int index, char ch)
    StringBuffer insert(int index, Object ob)
  - index specifies the index at which the string will be inserted
  - *StringBuffer sb=new StringBuffer("Hello ");*
    sb.insert(7,ök");

- **reverse()**
  - reverses the characters in a StringBuffer object
  - *StringBuffer reverse()*
  - *StringBuffer sb=new StringBuffer("Hello");*
    sb.reverse();

StringBuffer Functions...

StringBuffer Functions...

- **replace()**
  - replaces one set of characters with another set
  - *StringBuffer replace(int start, int end, String str)*
  - *StringBuffer sb=new StringBuffer("Bhubaneswar");*
    sb.replace(3,4,"va");
- **substring()**
  - returns a portion of a StringBuffer
  - *String substring(int start)*
    String substring(int start, int end)
- **delete()**
  - deletes a sequence of characters
  - *StringBuffer delete(int start, int end)*
  - *sb.delete(2,4);*
- **deleteCharAt()**
  - deletes the character at the pos index
  - *StringBuffer deleteCharAt(int pos)*
  - *sb.deleteCharAt(0);*

StringBuffer Functions...

StringBuffer Functions...

- **indexOf(String str)**
  - *int i = sb.indexOf(two");*
- **indexOf(String str, int start)**
  - *int i = sb.indexOf(two", 4);*
- **lastIndexOf(String str)**
  - *int i = sb.lastIndexOf(two");*
- **lastIndexOf(String str, int start)**
  - *int i = sb.lastIndexOf(two",4);*
- **trimToSize(int size)**
  - *sb.trimToSize(10);*