Programming in Java

Dr. Nyein Aye Maung Maung
Dr. Eng (Ritsumeikan University, Japan)
Lecturer
Computer Engineering and Information Technology Dept.
Yangon Technological University

Course Schedule

Week	Topics
Week 1	Overview of JAVA, Data Types, Variables and Arrays
Week 2	Operators and Control Statements
Week 3	Classes and A closer look at Methods and Classes
Week 4	Inheritance, Polymorphism, Abstraction and Encapsulation
Week 5	Packages and Interfaces
Week 6	Exception Handling and Multi-threaded Programming
Week 7	String Handling
Week 8	Exploring java.lang and More utilities classes
Week 9	Java Collections Framework
Week 10	Java I/O
Week 11	Basic Graphical User Interface
Week 12	Event Handling
Week 13	Database Programming
Week 14	Applet and Networking

Lecture 7

String Handling

Outline of Class (Lecture 7)

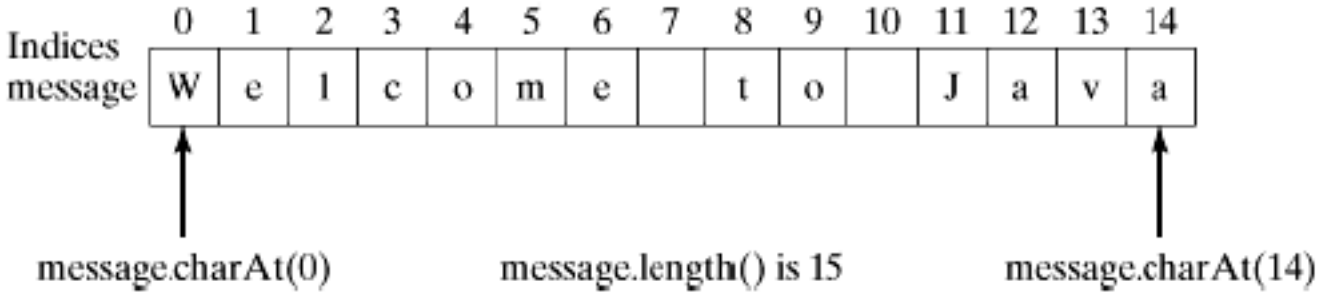
- String Handling
  - String constructors
  - String Comparison
  - Special String operations
  - Character Extraction
  - Obtaining substring
  - Searching Strings
  - Character Class
  - String Buffer
  - String Builder
  - String Tokenizer

Lecture Objectives

- To use the **String** class to process fixed strings
- To construct strings
- To use the **Character** class to process a single character
- To use the **StringBuilder** and **StringBuffer** classes to process flexible strings
- To distinguish among the **String**, **StringBuilder**, and **StringBuffer** classes

String

- Java string is a sequence of characters. They are objects of type **String**.
  - C Programming → char array
  - Java Programming → String object
- Once a String object is created it cannot be changed. To get changeable strings, use the class called StringBuffer or StringBuilder.



Creating a String Object

1. Implicit construction by assigning a string literal or
   - Java String literal is created by using double quotes.

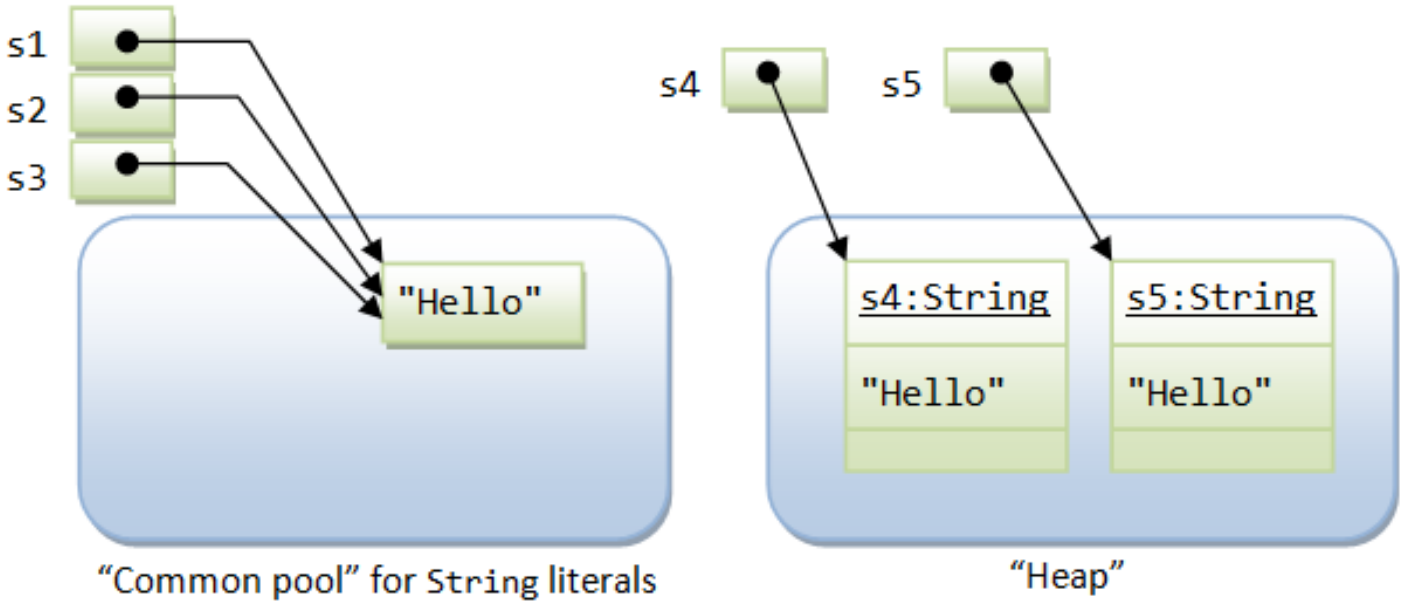
     String s="Welcome";

2. Explicitly creating a String object via the new operator and constructor
   - Java String is created by using a keyword "new".

     String s=new String("Welcome");

Cont’

```
String s1 = "Hello";                // String literal
String s2 = "Hello";                // String literal
String s3 = s1;                     // same reference
String s4 = new String("Hello");    // String object
String s5 = new String("Hello");    // String object
```



String Constructors

- Default constructor
  - **String s= new String();** //create an empty string
- Constructor with parameters initialized
  - char chars[]={‘a’, ‘b’, ‘c’};
  - **String s=new String(chars);** //create a string from a char array
  - **String s3=new String(s);** // create a string from an existing string

```
// Construct one String from another.
class MakeString {
  public static void main(String args[]) {
    char c[] = {'J', 'a', 'v', 'a'};
    String s1 = new String(c);
    String s2 = new String(s1);

    System.out.println(s1);
    System.out.println(s2);
  }
}
```

cont'

- String s= new String(char chars[], int startIndex, int numChars);

  //create a string as a subrange of char array

  ```
  char chars[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
  String s = new String(chars, 2, 3);
  ```

  s = cde

- String Literal

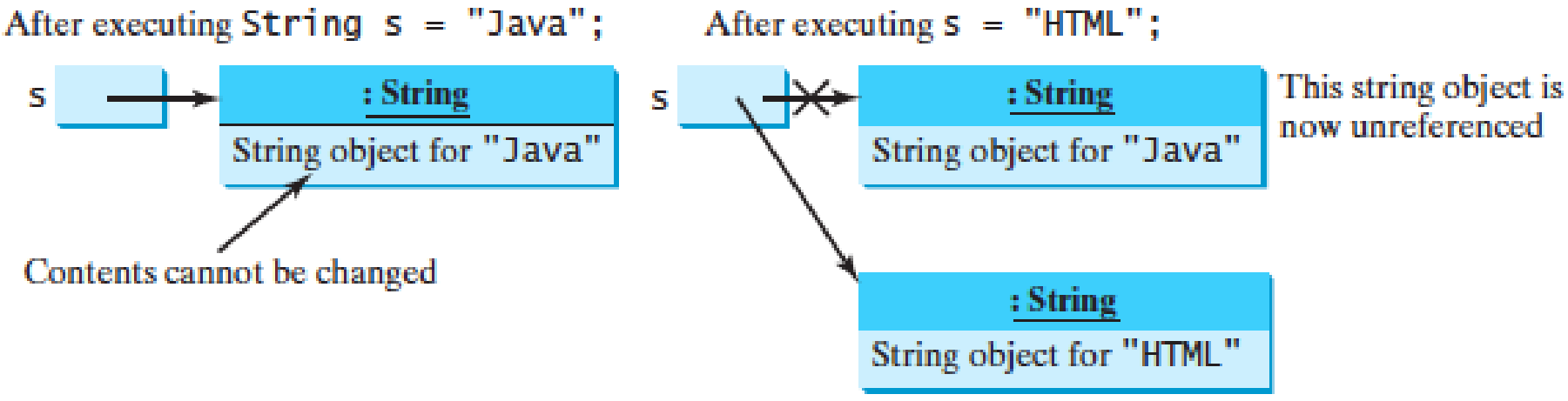
  - String s="How are you?";

Immutable Strings and Interned Strings

- A String object is immutable; its contents cannot be changed.

eg. String s = "Java";

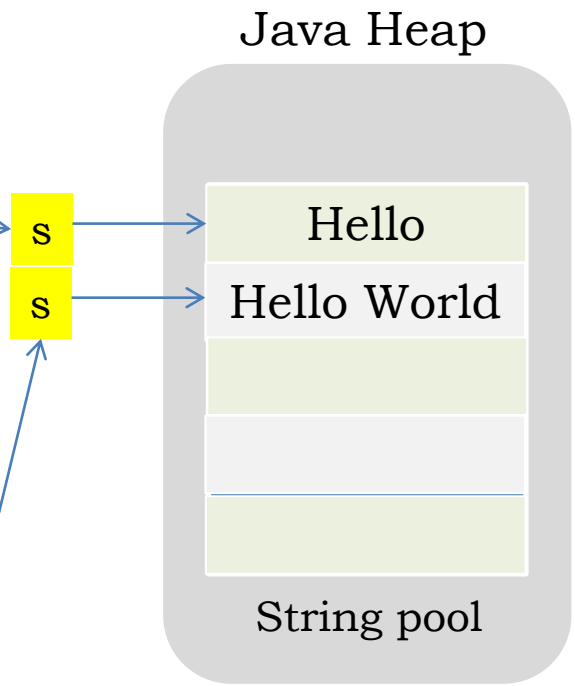
s = "HTML"; //create a new instance when the contents are changed.



- Sample program to test immutable

```
class Testimmutablestring{
 public static void main(String args[]){
   String s="Hello";
   s.concat(" World");//concat() method appends
   the string at the end, but s still refers to Hello
   System.out.println(s);
 }
}
```

Output
Hello

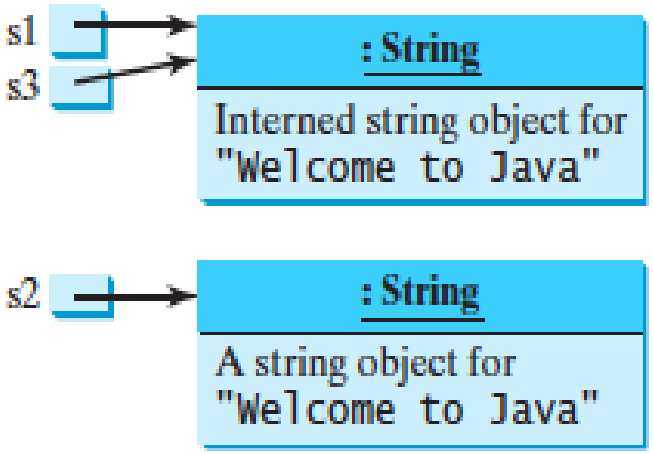



```
class Testimmutablestring{
 public static void main(String args[]){
   String s="Hello";
   s=s.concat(" World");//concat() method appends
   the string at the end
   System.out.println(s);
 }
}
```

Output
Hello World

- Interned String
  - Since strings are immutable and are ubiquitous in programming, the JVM uses a unique instance for string literals with the same character sequence in order to improve efficiency and save memory.
  - Such an instance is called an **interned string**

```
String s1= "Welcome to Java";
String s2= new String("Welcome to Java");
String s3= "Welcome to Java";

System.out.println("s1==s2 is"+ (s1==s2));
System.out.println("s1==s2 is"+ (s1==s3));
```



Display

```
s1==s2 is false
s1==s3 is true
```

String Comparison

- **boolean equals(String str)** :Compares two strings for equality (compare the contents of two strings).

```
String s1="Hello";
String s2=new String("Hello");
System.out.println("string1 and string2 have the same contents"+
(s1.equals(s2))); //output: True
```

- **boolean equalsIgnoreCase(String str)-** Compares two strings ignoring case

String Comparison

- **== operator** : Compare object reference (checks only whether string1 and string2 refer to the same object)

```
String s1="Hello";
String s2=new String("Hello");
System.out.println("s1 and s2 are the same object"+(s1==s2));
//output: False
```

- **compareTo(String s) method:** Compare values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string
  - s1 == s2 :0
  - s1 > s2 :positive value
  - s1 < s2 :negative value
- **compareToIgnoreCase(String s)**: Same as compareTo except that the comparison is case insensitive

- Sample programs on String Comparison #1

```
class equalsDemo {
public static void main(String args[]) {
        String s1 = "Hello";
        String s2 = "Hello";
        String s3 = "Good-bye";
        String s4 = "HELLO";
        System.out.println(s1 + " equals " + s2 + " -> " + s1.equals(s2));
        System.out.println(s1 + " equals " + s3 + " -> " + s1.equals(s3));
        System.out.println(s1 + " equals " + s4 + " -> " + s1.equals(s4));
        System.out.println(s1 + " equalsIgnoreCase " + s4+ "->"+s1.equalsIgnoreC
        ase(s4));
            }
            }
```

Output

```
Hello equals Hello -> true
Hello equals Good-bye -> false
Hello equals HELLO -> false
Hello equalsIgnoreCase HELLO->true
```

- Sample programs on String Comparison #2

```
class EqualsNotEqualTo {
        public static void main(String args[]) {
        String s1 = "Hello";
        String s2 = new String(s1);
        String s3 = “Hello”; or String s3=s1;
        System.out.println(s1 + " equals " + s2 + " -> " +
        s1.equals(s2));
        System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));
        System.out.println(s1+”==“+s3+" -> " + (s1 == s3));
        }
}
```

Output

Hello equals Hello -> true
Hello == Hello -> false
Hello==Hello -> true

- Sample programs on String Comparison #3

```
class EqualsNotEqualTo {
public static void main(String args[]) {
   String s1="Programming";
   String s2="Programming";
   String s3="Python";
   System.out.println(s1.compareTo(s2));
   System.out.println(s1.compareTo(s3));
   System.out.println(s3.compareTo(s1));
      }
}
```

Output

```
0
-7
7
```

More on String Comparison

- **regionMatches( )-** Compares specific region inside a string with another specific region in another string

  boolean regionMatches(int startIndex, String str2, int str2startIndex, int numChars)

  boolean regionMatches(boolean ignoreCase, int startIndex, String str2, int str2startIndex, int numChars)

```
String s = "This is a demo of the getChars method.";
String ss="Who is";
System.out.println(s.regionMatches(true, 5, ss,4,2)); //true
```

- **startsWith( ) –** Tests if this string starts with the specified prefix.

  public boolean startsWith(String prefix)

  eg. "Figure".startsWith("Fig"); **// true**

- **endsWith( ) -** Tests if this string ends with the specified suffix.

  public boolean endsWith(String suffix)

  eg. "Figure".endsWith("re"); **// true**

Example 7.1. Write a program that sorts an array of String using bubble sort.

```
class SortString {
    static String arr[] = {
    "Now", "is", "the", "time", "for", "all", "good", "men", "to", "come", "to",
    "the", "aid", "of", "their", "country"};
    public static void main(String args[]) {
            for(int j = 0; j < arr.length; j++) {
                    for(int i = j + 1; i < arr.length; i++) {
                            if(arr[i].compareTo(arr[j]) < 0) {
                                    String t = arr[j];
                                    arr[j] = arr[i];
                                    arr[i] = t;
                                    }
                            }
                    System.out.println(arr[j]);
}
}
}
```

Output

```
Now
aid
all
come
country
for
good
is
men
of
the
the
their
time
to
to
```

Getting String Length

- **String Length**
  - length() : return the number of characters in a string

```
System.out.println("Hello".length( )); //output=5
String s=new String("How are you?");
int ss=s.length();
System.out.println(ss); //output=12
```

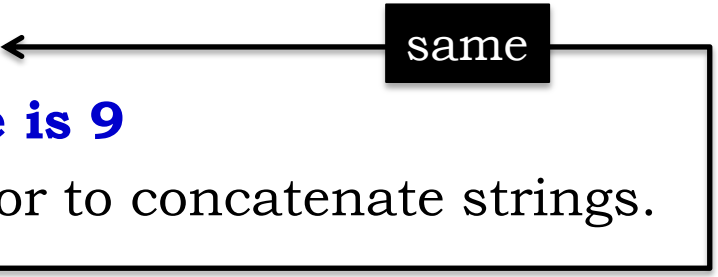
Combining Strings

- String Concatenation
  - Can use **concat**(String s) method to combine strings

Eg. String age="He is ";
String old="9"
String s=**age.concat(old);**
System.out.println(s); **//He is 9**

- Java allows to use **+** operator to concatenate strings.

String s=**age+old;**
System.out.println(s); **//He is 9**
String s2=s+ " years old.";
System.out.println(s2); **//He is 9 years old.**



cont'

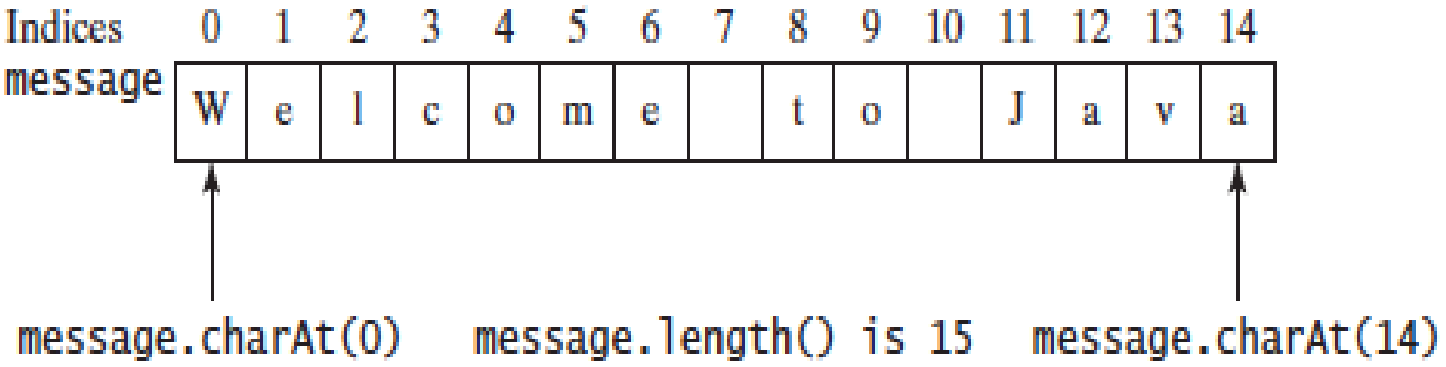
- String Concatenation with other data types

```
int age=9;
String s="He is "+ age + " years old.";
System.out.println(s); //output the same result

String s="four:"+2+2;
System.out.println(s); // four: 22

String s="four:"+(2+2);
System.out.println(s); // four: 4
```

Character Extraction

- **charAt( ):** Returns the character at the specified index. Index range from **0 to length( ) -1**

```
String s="Welcome to Java";
char ch = s.charAt(1); //ch=e;
```



Obtaining Substrings

- getChars() - Copies characters from this string into the destination character array.

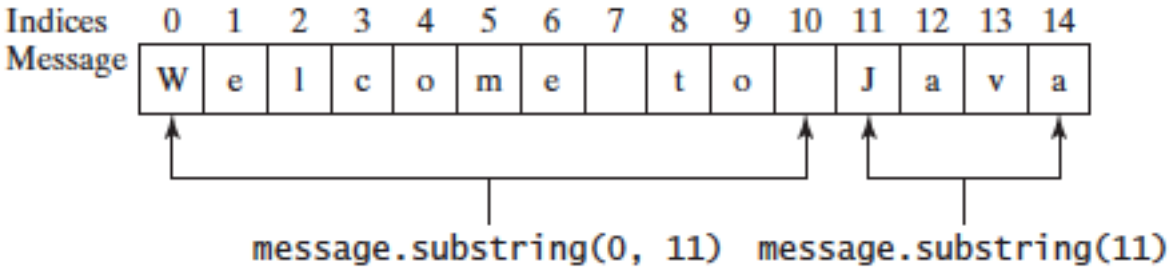
  void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)

  - srcBegin - index of the first character in the string to copy.
  - srcEnd - **index after the last character** in the string to copy.
  - dst - the destination array.
  - dstBegin - the start offset in the destination array.

- subString( ): Returns **a new string** that is a substring of this string.
  - String subString(int startIndex) //from startIndex to the end of string
  - String substring(int beginIndex, int endIndex) // From beginIndex through endIndex-1

    "smiles".substring(1, 5) // "mile“

    "unhappy".substring(2) // "happy”



Example 7.2. Demonstrate obtaining substring using getChars( ).

```
class getCharsDemo {
    public static void main(String args[]) {
    String s = "This is a demo of the getChars method.";
    int start = 10;
    int end = 14;
    char buf[] = new char[end - start];
    s.getChars(start, end, buf, 0);
    System.out.println(buf);
}
}
```

Output
demo

Modifying a String

- **replace( ):** Returns a new string that replaces all matching characters in this string with the new character.

  String replace(char original, char replacement)
  String replace(String subString1, String subString2)
  "hello".replace('l','w'); // "Hewwo"
  "This is".replece("is", "was"); // "Thwas was"

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

  String s = " Hi Mom! ".trim(); // "Hi Mom!"

- **toUpperCase():** Returns a new string with all characters converted to uppercase.

  "Welcome". toUpperCase() returns a new string, WELCOME.

- **toLowerCase():** Returns a new string with all characters converted to lowercase.

  "Welcome". toLowerCase() returns a new string, welcome.

cont’

- **split( ):** Returns an array of strings consisting of the substrings split by the delimiter.

```
String[] tokens = "Java#HTML#Perl".split("#");
```

Searching String

java.lang.String	
`+indexOf(ch: char): int`	Returns the index of the first occurrence of `ch` in the string. Returns `-1` if not matched.
`+indexOf(ch: char, fromIndex: int): int`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns `-1` if not matched.
`+indexOf(s: String): int`	Returns the index of the first occurrence of string `s` in this string. Returns `-1` if not matched.
`+indexOf(s: String, fromIndex: int): int`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns `-1` if not matched.
`+lastIndexOf(ch: int): int`	Returns the index of the last occurrence of `ch` in the string. Returns `-1` if not matched.
`+lastIndexOf(ch: int, fromIndex: int): int`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns `-1` if not matched.
`+lastIndexOf(s: String): int`	Returns the index of the last occurrence of string `s`. Returns `-1` if not matched.
`+lastIndexOf(s: String, fromIndex: int): int`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns `-1` if not matched.

Demonstrate the use of indexOf( ) and lastIndexOf( )

"Welcome to Java".indexOf('W') returns **0**.

"Welcome to Java".indexOf('o') returns **4**.

"Welcome to Java".indexOf('o', 5) returns **9**.

"Welcome to Java".indexOf("come") returns **3**.

"Welcome to Java".indexOf("Java", 5) returns **11**.

"Welcome to Java".indexOf("java", 5) returns **-1**.

"Welcome to Java".lastIndexOf('W') returns **0**.

"Welcome to Java".lastIndexOf('o') returns **9**.

"Welcome to Java".lastIndexOf('o', 5) returns **4**.

"Welcome to Java".lastIndexOf("come") returns **3**.

"Welcome to Java".lastIndexOf("Java", 5) returns **-1**.

"Welcome to Java".lastIndexOf("Java") returns **11**.

String Conversion

- Conversion between Strings and Arrays
  - **toCharArray( ) :** to convert all the characters in a String obj into a character array

    String s="Hello";

    char buf[ ]= s.toCharArray();
  - **getChars( )**

    char[] dst = {'J', 'A', 'V', 'A', '1', '3', '0', '1'};

    "CS372089".getChars(2, 6, dst, 4);

  Thus, dst becomes {'J', 'A', 'V', 'A', '3', '7', '2', '0'}.
  - **valueOf( )**: to convert an array of characters into a string

    char buf[ ]={'h','e','l','l','o'}

    String s=String.valueOf(buf);

Converting Characters and Numeric Values to Strings

java.lang.String	
+valueOf(c: char): String	Returns a string consisting of the character c.
+valueOf(data: char[]): String	Returns a string consisting of the characters in the array.
+valueOf(d: double): String	Returns a string representing the double value.
+valueOf(f: float): String	Returns a string representing the float value.
+valueOf(i: int): String	Returns a string representing the int value.
+valueOf(l: long): String	Returns a string representing the long value.
+valueOf(b: boolean): String	Returns a string representing the boolean value.

Example:

double **d**=0.0897; int i=89;

String s=String.valueOf(**d**); //double to string

String s=String.valueOf(**i**); //int to string

Object Class and toString

- Every class in Java is descended from the java.lang.Object class
- String and StringBuilder are implicitly subclasses of Object
- In default, toString( ) method is provided by Object class.

public String toString()

- Invoking toString() on an object returns a string that describes the object.
  - Default → classname@object's memory address in hexadecimal

```
Loan loan = new Loan();
System.out.println(loan.toString());
```

Output something likes **Loan@15037e5**

  - Not informative
  - Override the toString method to return a descriptive string representation of the object

Example 7.3. Override toString( ) method of Box class to display dimensions of the box.

```
// Override toString() for Box class.
class Box {
  double width;
  double height;
  double depth;
  Box(double w, double h, double d) {
       width = w;
       height = h;
       depth = d;
       }
@Override
public String toString() {
return "Dimensions are " + width + " by " +
depth + " by " + height + ".";
}
}
```

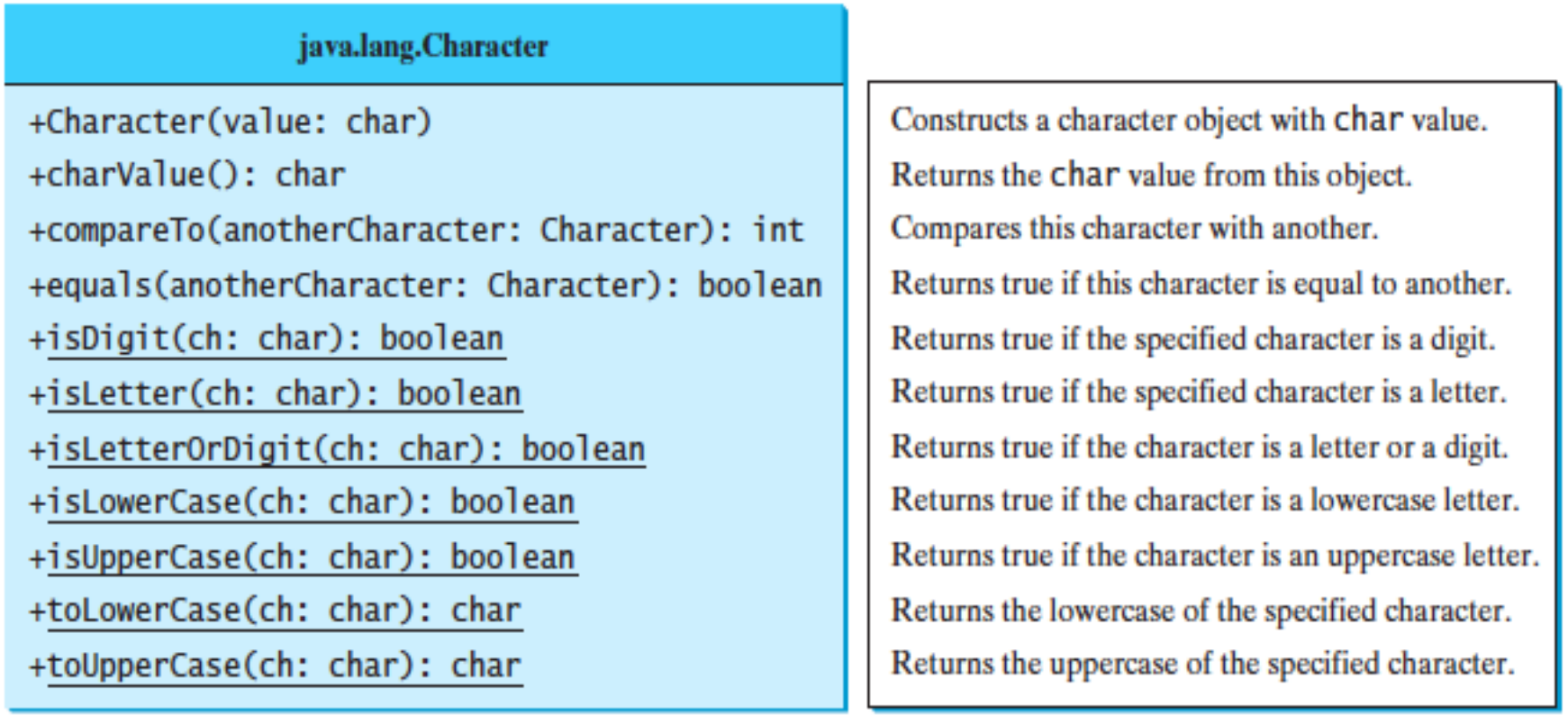
```
class toStringDemo {
   public static void main(String args[]) {
       Box b = new Box(10, 12, 14);
       System.out.println(b);
       String s = "Box b: " + b;
       System.out.println(s);
       }
}
```

```
Dimensions are 10.0 by 14.0 by 12.0
Box b: Dimensions are 10.0 by 14.0 by 12.0
```

The Character Class

- Create an object for a character using the Character class.
- A Character object contains a character value.

```
Character character = new Character('a');
```



Example 7.3. Write a program that prompts the user to enter a string and counts the number of occurrences of each letter in the string regardless of case.

```
import java.util.Scanner;

public class CountEachLetter {
  /** Main method */
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a string
    System.out.print("Enter a string: ");
    String s = input.nextLine();

    // Invoke the countLetters method to count each letter
    int[] counts = countLetters(s.toLowerCase());

    // Display results
    for (int i = 0; i < counts.length; i++) {
      if (counts[i] != 0)
        System.out.println((char)('a' + i) + " appears  " +
          counts[i] + ((counts[i] == 1) ? " time" : " times"));
    }
  }
```

```
/** Count each letter in the string */
  public static int[] countLetters(String s) {
    int[] counts = new int[26];

    for (int i = 0; i < s.length(); i++) {
      if (Character.isLetter(s.charAt(i)))
        counts[s.charAt(i) - 'a']++;
    }

    return counts;
  }
}
```

Output

```
Enter a string:abababx
a appears 3 times
b appears 3 times
x appears 1 time
```

String Builder and String Buffer

- StringBuilder and StringBuffer classes are similar to the String class except that the String class is immutable
- More flexible than String
  - Can add, insert, or append new contents into StringBuilder and StringBuffer objects
- The StringBuilder class is similar to StringBuffer except that the methods for modifying the buffer in StringBuffer are *synchronized*
  - Only one task is allowed to execute the methods with StringBuffer
  - Use StringBuffer if the class might be accessed by multiple tasks concurrently
  - Using StringBuilder is more efficient if it is accessed by just a single task

StringBuffer

- A StringBuffer is like a String, but can be modified.
- The length and content of the StringBuffer sequence can be changed through certain method calls.
- StringBuffer defines four constructors:
  - StringBuffer() //reserve room for 16 characters
  - StringBuffer(int size) //size defines size of StringBuffer
  - StringBuffer(String *str*) //rooms for str + reserve rooms for 16 additional char
  - StringBuffer(CharSequence *chars*) //rooms for chars + reserve rooms for 16 additional char

Methods for Modifying StringBuffer

- **length ( )** : length of a StringBuffer
- **capacity ( )** : total allocated capacity of a StringBuffer

  StringBuffer sb=new StringBuffer(40);
  StringBuffer sb2=new StringBuffer( );
  sb.lenght( ) → 0 ; sb2.length → 0;
  sb.capacity( ) → 40; sb2.capacity → 16;
  StringBuffer sb3=new StringBuffer("Hello");
  sb3.lenght( ) → 5 ; sb3.capacity( ) → 21;

- **ensureCapacity(int capacity)**: to preallocate room for a certain number of characters

  StringBuffer.ensureCapacity(int minCapacity);
  minCapacity <= Oringnal Capacity → Orininal capacity
  minCapacity > Oringnal Capacity && < (Original*2) +2 →(Original*2) +2
  minCapacity > (Original*2) +2 → minCapacity

Cont’

- **setLength(int length)**: Set the length of the buffer
- **charAt(int index)** : Index of char to be obtained
- **setCharAt(int index)**: Index of char being set //replace orinignal data
- **getChars**(int sourceStart, int sourceEnd, char target[], int targetStart)
- **append( )** : Concatenate the string representation of any other types of data to the end of the invoking StringBuffer object
  - StringBuffer append(String str); StringBuffer append(int num)
  - StringBuffer append(Object obj)

Example 7.4. Demonstrate append().

```
class setCharAtDemo {
public static void main(String args[]) {
    StringBuffer sb = new StringBuffer("Hello");
    System.out.println("buffer before = " + sb);
    System.out.println("charAt(1) before = " + sb.charAt(1));
    sb.setCharAt(1, 'i');
    sb.setLength(2);
    System.out.println("buffer after = " + sb);
    System.out.println("charAt(1) after = " + sb.charAt(1));
}
}
```

```
buffer before = Hello
charAt(1) before = e
buffer after = Hi
charAt(1) after = i
```

Example 7.5. Demonstrate charAt() and setCharAt().

```
class appendDemo {
public static void main(String args[]) {
    String s;
    int a = 42;
    StringBuffer sb = new StringBuffer(40);
    s = sb.append("a = ").append(a).append("!").toString();
    System.out.println(s);
}
}
```

```
a = 42!
```

- **insert( )** : insert one string into another
  - StringBuffer insert(int index, String str)
  - StringBuffer insert (int index, int num)
  - StringBuffer insert (int index, Object obj)
- **reverse( )** : to reverse a string

  StringBuffer reverse(String str)
- **delete ( ) & deleteCharAt( )**

  StringBuffer delete(int startIndex, int **endIndex**)

  //Through startIndex to endIndex-1

  StringBuffer deleteCharAt(int loc)
- **replace ( )** : to replace one set of characters with another set

  StringBuffer replace(int startIndex, int endIndex, String str)

  //Through startIndex to endIndex-1
- **substring( )** : to obtain a portion of a string
  - String substring(int *startIndex*) //startIndex through end of string
  - String substring(int *startIndex*, int *endIndex*) //Through startIndex to endIndex-1

Example 7.5. Demonstrate StringBuffer Methods.

```
public static void main (String args[]){
StringBuffer sb = new StringBuffer("Hello");
      System.out.println("Length"+sb.length()); // 5
      System.out.println("Capacity"+sb.capacity()); // 21 (16 rooms
                                                          reserve)
      sb.ensureCapacity(10);
      System.out.println("Capacity"+ sb.capacity());
      System.out.println("Character:"+sb.charAt(1)); // e
      sb.setCharAt(1,'i');
      System.out.println("After set:"+sb); // Hillo
      sb.setLength(2); // Hi
      System.out.println("After setting lenght:"+sb);
      sb.append("1").append("1").append(10);
      System.out.println("Append:"+sb);  // Hill10
      sb.insert(0, "Big "); // Big Hill10
```

```
        System.out.println("Insert:"+sb); // Big Hill10
        sb.delete(8,10);
         System.out.println("Delete: :"+sb); //Big Hill
        sb.replace(0, 3, "Small");
        System.out.println("Replace:"+sb); // Small Hill
        sb.reverse();
        System.out.println("Reverse: " +sb);//lliH llamS

}
}
```

Quiz Time !!!

- Quiz 1. Suppose that s1 , s2 , s3 , and s4 are four strings, given as follows:
  - String s1 = "Welcome to Java";
  - String s2 = s1;
  - String s3 = new String("Welcome to Java");
  - String s4 = "Welcome to Java";
- What are the results of the following expressions?

a) s1 == s2
b) b. s2 == s3
c) s1.equals(s2)
d) s2.equals(s3)
e) s1.compareTo(s2)
f) s2.compareTo(s3)
g) s1 == s4
h) s1.charAt(0)
i) s1.indexOf('j')
j) s1.indexOf("to")
k) s1.lastIndexOf('a')
l) s1.lastIndexOf("o", 15)
m) s1.length()
n) s1.startsWith("Wel")
o) s1.endsWith("Java")

Quiz 2.

```
class indexOfDemo {
    public static void main(String args[]) {
   String s = "Now is the time for all good men " +
   "to come to the aid of their country.";
System.out.println(s);
System.out.println("indexOf(t) = " + s.indexOf('t'));
System.out.println("lastIndexOf(t) = " + s.lastIndexOf('t'));
System.out.println("indexOf(the) = " + s.indexOf("the"));
System.out.println("lastIndexOf(the) = " + s.lastIndexOf("the"));
System.out.println("indexOf(t, 10) = " + s.indexOf('t', 10));
System.out.println("lastIndexOf(t, 60) = " + s.lastIndexOf('t', 60));
System.out.println("indexOf(the, 10) = " + s.indexOf("the", 10));
System.out.println("lastIndexOf(the, 60) = " +s.lastIndexOf("the” 60));
}
}
```

- Quiz 3. Suppose that s1 and s2 are two strings. Which of the following statements or expressions are incorrect?

```
String s = new String("new string");
String s3 = s1 + s2;
String s3 = s1 - s2;
s1 == s2;
s1 >= s2;
s1.compareTo(s2);
int i = s1.length();
char c = s1(0);
char c = s1.charAt(s1.length());
```

Quiz 4.

Suppose that s1 and s2 are given as follows:

StringBuilder s1 = new StringBuilder("Java");

StringBuilder s2 = new StringBuilder("HTML");

Show the value of s1 after each of the following statements. Assume that the statements

are independent.

a. s1.append(" is fun");

b. s1.append(s2);

c. s1.insert(2, "is fun");

d. s1.insert(1, s2);

e. s1.charAt(2);

f. s1.length();

g. s1.deleteCharAt(3);

h. s1.delete(1, 3);

i. s1.reverse();

j. s1.replace(1, 3, "Computer");

k. s1.substring(1, 3);

l. s1.substring(2);

Quiz 5

What will be the value of str after the following statements are executed?

```
String str = "Dr. Decaffeinated";
StringBuffer strBuf = new StringBuffer(str.substring(6, 10) );
strBuf.setCharAt(1, 'o');
strBuf.append( 'e' ).append('e');
str = strBuf.toString( );
```

What is the result of the following?

```
String stringA = " Wild " ;
String stringB = " Irish ";
String stringC = " Rose ";
String result  = stringA.trim() + stringB + stringC.trim();
```

Quiz 6

Which of these class is superclass of String and StringBuffer class?

a) java.util
b) java.lang
c) ArrayList
d) None of the mentioned

What is the output?

```
public class Test{
        public static void main(String args[]){
                String s1 = new String("Hello");
                  String s2 = new String("Hellow");
                  System.out.println(s1 = s2);
          }
}
```

Quiz 7

Which of the followings is correct?

```
public class Test{
        public static void main(String args[]){
                String s1 = 'SITHA' ;
                String s2 = 'RAMA';
                System.out.println(s1.charAt(0) > s2.charAt(0));
        }
}
```

A.True
B. false
C. 0
D. Compilation error
E. Throws Exception

Quiz 8

String str 1 = "Kolkata".replace('k', 'a');
What is the result in str1?

How many objects will be created?
String a = new String("Examveda");
String b = new String("Examveda");
String c = "Examveda";
String d = "Examveda";

What will be output?
String S1 = "S1 ="+ "123"+"456";
String S2 = S1+(123+456);

Exercises on String

1. Write a method that counts the number of letters in a String using the following header:

   public static int countLetters(String s)

Write a test program that prompts the user to enter a string and displays the number of letters in the String.

```
public class CountLetterTest {

  public static void main(String[] args) {

    // Prompt the user to enter a string
    java.util.Scanner input = new java.util.Scanner(System.in);
    System.out.print("Enter a string: ");
    String s = input.nextLine();
    System.out.println("The number of letters is " + countLetters(s));

  }
```

//method for counting letters

```
public static int countLetters(String s) {

  int count = 0;

  for (int i = 0; i < s.length(); i++) {

    if (Character.isLetter(s.charAt(i))) {

      count++;

    }
  }
  return count;
}

}
```

2. Write a program that output the name of months beginning with the letter “J” and those end with “ber”.

```
public class findMonths{
    public static void main(String args[])
    {
            String[] month = {"January","February","March","April","May","June",
                              "July", "August","September","October","November",
                              "December"};
            System.out.println("Months start with J:");
            for(int i=0;i<month.length;i++)
            {
                    if(month[i].startsWith("J"))
                    System.out.println(month[i]);
            }
            System.out.println("Months end with 'ber':");
            for(int j=0;j<month.length;j++)
            {
                    if(month[j].endsWith("ber"))
                    System.out.println(month[j]);
            }
}
}
```

3. Write a method that checks whether two words are anagrams. Two words are anagrams if they contain the same letters in any order. For example, "silent" and "listen" are anagrams. The header of the method is as follows:

public static boolean isAnagram(String s1, String s2)

Write a main method to invoke isAnagram("silent", "listen"), isAnagram("garden", "ranged"), and isAnagram("split", "lisp").

```
public class anagram {
        public static void main(String args[])
         System.out.println( "silent and listen are" +
          (isAnagram("silent", "listen") ? " anagram." : " not anagram."));
         System.out.println( "garden and ranged are" +
          (isAnagram("garden", "ranged") ? " anagram." : " not anagram."));
         System.out.println( "split and lisp are" +
          (isAnagram("split", "lisp") ? " anagram." : " not anagram."));

        }
```

//Algorithm to check whether the String is Anagram or not

```
public static boolean isAnagram(String s1, String s2) {
 if (s1.length() != s2.length()) return false;

      String newS1 = sortString(s1);
      String newS2 = sortString(s2);

 for (int i = 0; i < newS1.length(); i++) {
    if (newS1.charAt(i) != newS2.charAt(i))
      return false;
   }

    return true;
  }
```

//Algorithm to sort string

```
public static String sortString(String s)
{
        String sorted=new String();
        char[] ch=s.toCharArray();
        for(int j = 0; j < ch.length; j++) {
                for(int i = j + 1; i < ch.length; i++) {
                  if(ch[i]<ch[j]) {
                                        char t = ch[j];
                                        ch[j] = ch[i];
                                        ch[i] = t;
                                }
                        }
        }
                sorted=String.valueOf(ch);
               return sorted;
        }
}
```

4. Write a program that prompts the user to enter a string and reports whether the string is a Palindrome or not. A string is a palindrome if it reads the same forward and backward. The words “mom,” “dad,” and “noon,” for instance, are all palindromes.

```
import java.util.Scanner;
public class palindrome {
/** Main method */
    public static void main(String[] args) {
    // Prompt the user to enter a string;
    Scanner s=new Scanner(System.in);
    System.out.println("Enter  a string:");
    // Declare and initialize input string
    String inputString = s.nextLine();
     if (isPalindrome(inputString))
        inputString = inputString + " is a palindrome";
     else
        inputString = inputString + " is not a palindrome";
     // Display the result
     System.out.println(inputString);
     }
```

```
/** Method to check whether a string is a palindrome or not */
public static boolean isPalindrome(String s) {
        // The index of the first character in the string
        int low = 0;
        // The index of the last character in the string
        int high = s.length ()- 1;
        while (low < high) {
        if (s.charAt(low) != s.charAt(high))
        return false; // Not a palindrome
        low++;
         high--;
        }
        return true; // The string is a palindrome
        }
}
```

Assignments

1. Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows:

- A password must have at least eight characters.
- A password consists of only letters and digits.
- A password must contain at least two digits.

Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.

2. (Occurrences of each digit in a string) Write a method that counts the occurrences of each digit in a string using the following header: ublic static int[] count(String s) The method counts how many times a digit appears in the string. The return value is an array of ten elements, each of which holds the count for a digit. ‘

For example, after executing int[] counts = count("12203AB3") , counts[0] is 1 , counts[1] is 1 , counts[2] is 2 , and counts[3] is 2 . Write a test program that prompts the user to enter a string and displays the number of occurrences of each digit in the string.

3. Write a method that finds the number of occurrences of a specified character in the string using the following header:

public static int count(String str, char a)

For example, count("Welcome", 'e') returns 2.

Next Week Lecture

- Exploring java.lang and More utilities classes