Programming Language II
CSE-215

Prof. Dr. Mohammad Abu Yousuf
yousuf@juniv.edu

Java String

- In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s=new String(ch);
```

is same as:

```
String s="javatpoint";
```

- **Java String** class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

How to create a string object?

There are two ways to create String object:

- By string literal
- By new keyword

1) String Literal

- Java String literal is created by using double quotes. For Example:

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

- Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

2) By new keyword

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

- In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

Java String Example

```
1.public class StringExample{
2.public static void main(String args[]){
3.String s1="java";//creating string by java string literal
4.char ch[]={'s','t','r','i','n','g','s'};
5.String s2=new String(ch);//converting char array to string
6.String s3=new String("example");//creating java string by new keywor
  d
7.System.out.println(s1);
8.System.out.println(s2);
9.System.out.println(s3);
10.}}
```

Output:
java
strings
example

Java String class methods

- The java.lang.String class provides many useful methods to perform operations on sequence of char values.

Immutable String in Java

- In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.
- Once string object is created its data or state can't be changed but a new string object is created.

```
1.class Testimmutablestring{
2. public static void main(String args[]){
3.   String s="Saif";
4.   s.concat(" Shams");//concat() method appends the s
  tring at the end
5.   System.out.println(s);//will print Saif because strings
   are immutable objects
6. }
7.}
```

Output: Saif

Immutable String in Java

- if we explicitly assign it to the reference variable, it will refer to "Saif Shams" object. For example:

```
1.class Testimmutablestring1{
2. public static void main(String args[]){
3.   String s="Saif";
4.   s=s.concat(" Shams");
5.   System.out.println(s);
6. }
7.}
```

Output: Saif Shams

Java String compare

- We can compare string in java on the basis of content and reference.
- It is used in **authentication** (by equals() method), **sorting** (by compareTo() method), **reference matching** (by == operator) etc.
- There are three ways to compare string in java:
  - By **equals()** method
  - By **= =** operator
  - By **compareTo()** method

1) String compare by equals() method

- The String equals() method compares the original content of the string. It compares values of string for equality. String class provides two methods:

- **public boolean equals(Object another)** compares this string to the specified object.

- **public boolean equalsIgnoreCase(String another)** compares this String to another string, ignoring case.

1) String compare by equals() method

```
1.class Teststringcomparison1{
2. public static void main(String args[]){
3.   String s1="Sachin";
4.   String s2="Sachin";
5.   String s3=new String("Sachin");
6.   String s4="Saurav";
7.   System.out.println(s1.equals(s2));//true
8.   System.out.println(s1.equals(s3));//true
9.   System.out.println(s1.equals(s4));//false
10. }
11.}
```

Output: true
true
false

1) String compare by equals() method

```
1.class Teststringcomparison2{
2. public static void main(String args[]){
3.   String s1="Sachin";
4.   String s2="SACHIN";
5.
6.   System.out.println(s1.equals(s2));//false
7.   System.out.println(s1.equalsIgnoreCase(s2));//true
8. }
9.}
```

Output:

false
true

2) String compare by == operator

- The = = operator compares references not values.

```
1.class Teststringcomparison3{
2. public static void main(String args[]){
3.   String s1="Sachin";
4.   String s2="Sachin";
5.   String s3=new String("Sachin");
6.   System.out.println(s1==s2);//true (because both refer to sam
   e instance)
7.   System.out.println(s1==s3);//false(because s3 refers to insta
   nce created in nonpool)
8. }
9.}
```

Output: true
false

3) String compare by compareTo() method

- The String compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.
- Suppose s1 and s2 are two string variables. If:
  - **s1 == s2** :0
  - **s1 > s2** :positive value
  - **s1 < s2** :negative value

3) String compare by compareTo() method

```
1.class Teststringcomparison4{
2. public static void main(String args[]){
3.   String s1="Sachin";
4.   String s2="Sachin";
5.   String s3="Ratan";
6.   System.out.println(s1.compareTo(s2));//0
7.   System.out.println(s1.compareTo(s3));//1(because s1
  >s3)
8.   System.out.println(s3.compareTo(s1));//-1
  (because s3 < s1 )
9. }
10.}
```

```
Output: 0
        1
       -1
```

```java
public class Exercise5 {
public static void main(String[] args)
    {
        String str1 = "This is Exercise 1";
        String str2 = "This is Exercise 2";

        System.out.println("String 1: " + str1);
        System.out.println("String 2: " + str2);

        // Compare the two strings.
        int result = str1.compareTo(str2);

        // Display the results of the comparison.
        if (result < 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is less than " +
                "\"" + str2 + "\"");
        }
        else if (result == 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is equal to " +
                "\"" + str2 + "\"");
        }
        else // if (result > 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is greater than " +
                "\"" + str2 + "\"");
        }
    }
```

Write a java program to compare two strings lexicographically.

```
String 1: This is Exercise 1
String 2: This is Exercise 2
"This is Exercise 1" is less than "This is Exercise 2"
```

```java
public class Exercise6 {

public static void main(String[] args)
    {
        String str1 = "This is exercise 1";
        String str2 = "This is Exercise 1";

        System.out.println("String 1: " + str1);
        System.out.println("String 2: " + str2);

        // Compare the two strings.
        int result = str1.compareToIgnoreCase(str2);

        // Display the results of the comparison.
        if (result < 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is less than " +
                "\"" + str2 + "\"");
        }
        else if (result == 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is equal to " +
                "\"" + str2 + "\"");
        }
        else // if (result > 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is greater than " +
                "\"" + str2 + "\"");
        }
    }
```

```
String 1: This is exercise 1
String 2: This is Exercise 1
"This is exercise 1" is equal to
"This is Exercise 1"
```

Write a Java program to compare a given string to the specified character sequence.

```
public class Exercise9 {
public static void main(String[] args) {
    String str1 = "example.com", str2 = "Example.com";
    CharSequence cs = "example.com";
     System.out.println("Comparing "+str1+" and "+cs+": " + str1.contentEquals(cs));
      System.out.println("Comparing "+str2+" and "+cs+": " + str2.contentEquals(cs));
          }
}
```

```
Comparing example.com and example.com: true
Comparing Example.com and example.com: false
```

```
  public static void main(String[] args)
    {
        String columnist1 = "Stephen Edwin King";
        String columnist2 = "Walter Winchell";
        String columnist3 = "stephen edwin king";

        // Test any of the above Strings equal to one another
        boolean equals1 = columnist1.equalsIgnoreCase(columnist2);
        boolean equals2 = columnist1.equalsIgnoreCase(columnist3);

        // Display the results of the equality checks.
        System.out.println("\"" + columnist1 + "\" equals \"" +
            columnist2 + "\"? " + equals1);
        System.out.println("\"" + columnist1 + "\" equals \"" +
            columnist3 + "\"? " + equals2);
    }
}
```

```
"Stephen Edwin King" equals "Walter Winchell"? false
"Stephen Edwin King" equals "stephen edwin king"? true
```

String Concatenation in Java

- In java, string concatenation forms a new string *that is* the combination of multiple strings. There are two ways to concat string in java:
- By + (string concatenation) operator
- By concat() method

2) String Concatenation by concat() method

```
1.class TestStringConcatenation3{
2. public static void main(String args[]){
3.   String s1="Sachin ";
4.   String s2="Tendulkar";
5.   String s3=s1.concat(s2);
6.   System.out.println(s3);//Sachin Tendulkar
7.  }
8.}
```

Substring in Java

- A part of string is called **substring**. In other words, substring is a subset of another string. In case of substring ***startIndex*** is inclusive and ***endIndex*** is exclusive.
- We can get substring from the given string object by one of the two methods:
- **public String substring(int startIndex):** This method returns new String object containing the substring of the given string from specified **startIndex** (inclusive).
- **public String substring(int startIndex, int endIndex):** This method returns new String object containing the substring of the given string from specified **startIndex** to **endIndex.**

Substring in Java

```
1.String s="hello";
2.System.out.println(s.substring(0,2));//he
```

In the above substring, 0 points to h but 2 points to e (because end index is exclusive).

```
1.public class TestSubstring{
2. public static void main(String args[]){
3.   String s="SachinTendulkar";
4.   System.out.println(s.substring(6));//Tendulkar
5.   System.out.println(s.substring(0,6));//Sachin
6. }
7.}
```

Java String charAt()

- The **java string charAt()** method returns *a char value at the given index number*.
- The index number starts from 0 and goes to n-1, where n is length of the string. It returns **StringIndexOutOfBoundsException** if given index number is greater than or equal to this string length or a negative number.

Java String charAt() Example 1

```
1.public class CharAtExample{
2.public static void main(String args[]){
3.String name="javatpoint";
4.char ch=name.charAt(4);//returns the char value at the 4t
  h index
5.System.out.println(ch);
6.}}
```

Output: t

Java String charAt() Example 2

```
1.public class CharAtExample3 {
2.    public static void main(String[] args) {
3.    String str = "Welcome to Javatpoint portal";
4.    int strLength = str.length();
5.    // Fetching first character
6.    System.out.println("Character at 0 index is: "+ str.charAt(0));
7.    // The last Character is present at the string length-1 index
8.    System.out.println("Character at last index is: "+ str.charAt(strLe
   ngth-1));
9.    }
10.}
```

Output:

Character at 0 index is: W

Character at last index is: l

Java String charAt() Example 3

```
1.public class CharAtExample5 {
2.    public static void main(String[] args) {
3.        String str = "Welcome to Javatpoint portal";
4.        int count = 0;
5.        for (int i=0; i<=str.length()-1; i++) {
6.            if(str.charAt(i) == 't') {
7.                count++;
8.            }
9.        }
10.       System.out.println("Frequency of t is: "+count);
11.   }
12.}
```

Output:

Frequency of t is: 4

Write a Java program to get the character at the given index within the String.

```
public class Exercise1 {
    public static void main(String[] args)
     {
         String str = "Java Exercises!";
         System.out.println("Original String = " + str);
         // Get the character at positions 0 and 10.
         int index1 = str.charAt(0);
         int index2 = str.charAt(10);

         // Print out the results.
         System.out.println("The character at position 0 is " +
             (char)index1);
         System.out.println("The character at position 10 is " +
             (char)index2);
     }
}
```

```
Original String = Java Exercises!
The character at position 0 is J
The character at position 10 is i
```

Thank you