SNS COLLEGE OF ENGINEERING

Coimbatore-107
An Autonomous Institution

COURSE NAME : 23CSB201 & Object Oriented Programming

II YEAR/ III SEMESTER

UNIT – IV I/O, GENERICS, STRING HANDLING

Topic: String basics

Dr.P.Poonkodi

Assistant Professor(SG)

Department of Computer Science and Technology

Introduction

- a string is a sequence of characters
- For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o‘
- We use **double quotes** to represent a string in Java

```
// Create String
        String type = "Java programming";
```

- we have created a string variable named **type**
- The variable is initialized with the string **Java Programming**
- there is another way of creating Java strings (using the new keyword)
- all string variables are instances of the String class

Example

```
class Main
{
    public static void main(String[] args)
    {
            // create strings
            String first = "Java";
            String second = "Python";
            String third = "JavaScript";
            // print strings
            System.out.println(first);
            System.out.println(second);
            System.out.println(third);
     }
}
```

String class

- java.lang.Object
- java.lang.String
- It represents character strings
- All string literals are implemented as instances of this class
- Strings are constants
- String buffers support mutable strings, because string objects are immutable they can be shared
- Example

  String str="abc";
- class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase

String Class

- Java language provides special support for the string concatenation operator (+), and for conversion of other objects to strings
- String concatenation is implemented through the StringBuilder(or **StringBuffer**) class and its append method
- String conversions are implemented through the method toString

Creating Strings Using the New Keyword

- strings in Java are objects, we can create strings using the new keyword as well
- // create a string using the new keyword
- String name = new String("Java String");

Java String Operations

- various string methods to perform different operations on strings
- **length()**
- **compare()**
- contains()
- substring()
- join()
- replace()

Example

```
public class Main {
    public static void main(String[] args) {
        String str = "  Hello World  ";
        System.out.println(str.length());              // 15
        System.out.println(str.charAt(2));             // 'H'
        System.out.println(str.substring(2, 7));       // 'Hello'
        System.out.println(str.trim());                // 'Hello World'
        System.out.println(str.toLowerCase());         // '  hello world  '
        System.out.println(str.contains("World"));     // true
        System.out.println(str.replace("World", "Java"));  // '  Hello Java  ‘
```

Example

```
        String[] words = str.trim().split(" ");
        for (String word : words)
        {
            System.out.println(word);                    // Hello \n World
        }
    }
}
```

String Buffer Class

- StringBuffer class is used to create **mutable (modifiable) sequences of characters**
- Unlike String, which is **immutable** (once created, cannot be changed)
- StringBuffer allows you to modify the contents without creating new objects
- java.lang package

String Buffer Class

- StringBuffer sb = new StringBuffer(); // Empty buffer, default capacity 16
- StringBuffer sb = new StringBuffer("Hello"); // Initialized with a string
- StringBuffer sb = new StringBuffer(50); // Specified capacity

Example

```
public class Main
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("Hello");
        sb.append(" World");
        sb.insert(5, ",");
        sb.replace(0, 5, "Hi");
        sb.reverse();
        System.out.println(sb);  // Output: dlroW ,iH
    }
}
```

References

- Java : the complete Reference (Eleventh Edition), Herbert Schildt, 2018.


Thank You!