# Strings

Dr. H. B. Prajapati

Associate Professor
Department of Information Technology
Dharmsinh Desai University

9 July '20

Core Java Technology

## Table of contents

1 String class

2 StringBuffer class

3 StringTokenizer class

4 Command-Line Arguments

## String in Java

- A string is a sequence of characters.
- In many programming languages, a string is considered as an array of characters.
- In Java, a string is considered as an object of String class (i.e., java.lang.String).

## Outline of Presentation

1 String class

2 StringBuffer class

3 StringTokenizer class

4 Command-Line Arguments

## Creating a String Object

- Earlier, we have passed string literals to println() method.
- The Java compiler actually converts these string literals into String objects and passes to println() method.
- To create a string explicitly, we use the following syntax:

```
String s = new String("String in Java");
```

- Alternatively, we can declare and create a string using the following syntax:

```
String s = "String in Java";
```

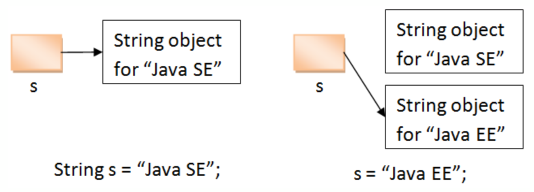
- We can also create a string from an array of characters:

```
char[] chars = {'S','t','r','i','n','g',' ',
    'i','n',' ','J','a','v','a'};
String s = new String(chars);
```

## String is Immutable

- A String object is immutable. That means, its contents cannot be changed.
- For the following code, a new string object is created when the second statement is executed:

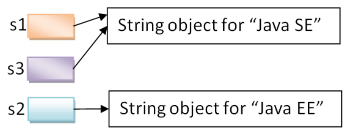
```
String s = "Java SE";
s = "Java EE";
```


String object for "Java SE"
s
String s = "Java SE";
String object for "Java SE"
s
String object for "Java EE"
s = "Java EE";

## Interned String

- As String is immutable, the JVM uses a unique instance for same string literals to save memory and improve efficiency.
- Such instance is called interned.

```
String s1 = "Java SE";
String s2 = new String("Java EE");
String s3 = "Java SE";
```



## String Comparison

- For comparing values of two variables of primitive type we use == comparison operator.
- But, since a string in Java is an object, two strings cannot be compared using == comparison operator.
- To compare two strings, Java provides equals() method in String class.
- The signature of the equals() method is as follows:

```
boolean equals(String);
```

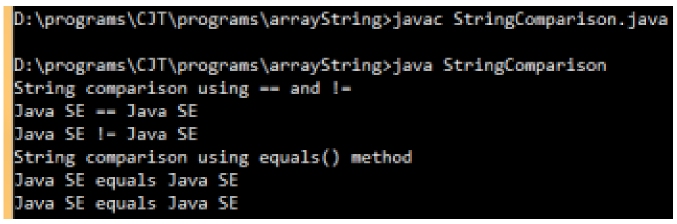
## Program: String Comparison, Slide - I

```
class StringComparison{
   public static void main(String[] args){
      String s1 = "Java SE";
      String s2 = "Java SE";
      String s3 = new String("Java SE");
      System.out.println("String comparison
         using == and !=");
      if(s1==s2)
         System.out.println(s1+" == "+s2);
      else
         System.out.println(s1+" != "+s2);
      if(s1==s3)
         System.out.println(s1+" == "+s3);
      else
         System.out.println(s1+" != "+s3);

      System.out.println("String comparison
         using equals() method");
```

## Program: String Comparison, Slide - II

```
      if(s1.equals(s2))
         System.out.println(s1+" equals "+s2);
      else
         System.out.println(s1+" !equals "+s2);
      if(s1.equals(s3))
         System.out.println(s1+" equals "+s3);
      else
         System.out.println(s1+" !equals "+s3);
   }
}
```

## Program: String Comparison, Slide - III



```
D:\programs\CJT\programs\arrayString>javac StringComparison.java

D:\programs\CJT\programs\arrayString>java StringComparison
String comparison using == and !=
Java SE == Java SE
Java SE != Java SE
String comparison using equals() method
Java SE equals Java SE
Java SE equals Java SE
```

## Ordering and Comparison using compareTo()

- To compare two string, we can also use compareTo() method, as shown below:

```
s1.compareTo(s2);
```

- The compareTo() returns 0 if two strings are equal. Note: equals() method returns boolean value.
- The compareTo() can be used to order strings in lexicographic (dictionary) order.
  - If s1 comes first (in lexicographic order) than s2, then compareTo() will return value less than 0
  - If s1 comes after s2 (in lexicographic order), then compareTo() will return value greater than 0.
  - The exact return value will be the difference between the first non-matching characters of s1 and s2.

## Program: Comparison and Ordering, Slide - I

```java
class StringCompareTo{
   public static void main(String[] args){
      String s1 = "Java SE";
      String s2 = "Java se";
      String s3 = new String("Java SE");
      String s4 = new String("Java EE");
      int result=0;

      System.out.println("String comparison
         using compareTo()");
      if(s1.compareTo(s3)==0)
         System.out.println("\t"+s1+" == "+s3);
      else
         System.out.println("\t"+s1+" != "+s3);

      System.out.print("Ordering strings using
         compareTo(), RV=");
      if((result=s1.compareTo(s2))<0){
```

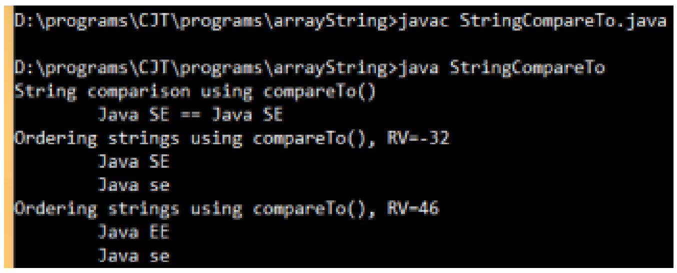
## Program: Comparison and Ordering, Slide - II

```java
         System.out.println(result);
         System.out.println("\t"+s1);
         System.out.println("\t"+s2);
      } else{
         System.out.println(result);
         System.out.println("\t"+s2);
         System.out.println("\t"+s1);
      }

      System.out.print("Ordering strings using
         compareTo(), RV=");
      if((result=s2.compareTo(s4))<0){
         System.out.println(result);
         System.out.println("\t"+s2);
         System.out.println("\t"+s4);
      } else{
         System.out.println(result);
         System.out.println("\t"+s4);
         System.out.println("\t"+s2);
```

## Program: Comparison and Ordering, Slide - III

```java
      }
   }
}
```

## Program: Comparison and Ordering, Slide - IV



```
D:\programs\CJT\programs\arrayString>javac StringCompareTo.java

D:\programs\CJT\programs\arrayString>java StringCompareTo
String comparison using compareTo()
        Java SE == Java SE
Ordering strings using compareTo(), RV=-32
        Java SE
        Java se
Ordering strings using compareTo(), RV=46
        Java EE
        Java se
```

## Other ways of String comparison

- To compare two strings, we can use compareTo()
- If we want to compare strings ignoring cases, we can use equalsIgnoreCase() method.
- To compare strings, we cannot use comparison operators such as `>, >=. <, <=, or !=.`

## String Methods

### Length, Characters, Combining, and Substring

```java
int length();
char charAt(int index);
String concat(String s2);
String substring(int start);
String substring(int start, int end);
```

- The length(); method returns the number of characters.
- The charAt(index); method returns the character present on the specified index.
- The concat(s2); method returns a new string that concatenates invoking string with s2.
- The substring(start); method returns a new string containing characters starting from start index to the last character.
- The substring(start, end); method returns a new string containing characters starting from start index to the end index-1.

## Program: Extract Information, Slide - I

```java
import java.util.Scanner;
class ExtractInformation{
   public static void main(String[] args){
      Scanner input=new Scanner(System.in);
      String s;
      String[] info;

      System.out.println("Format Ex: IT000-Harshad Prajapati");
      System.out.print("Enter profile name: ");
      s = input.nextLine();
      if(isValid(s)){
         printValidity(s);
         info=extractInfo(s);
         System.out.println("Extracted Information of "+s);
         System.out.println("\tBranch: "+info[0]);
```

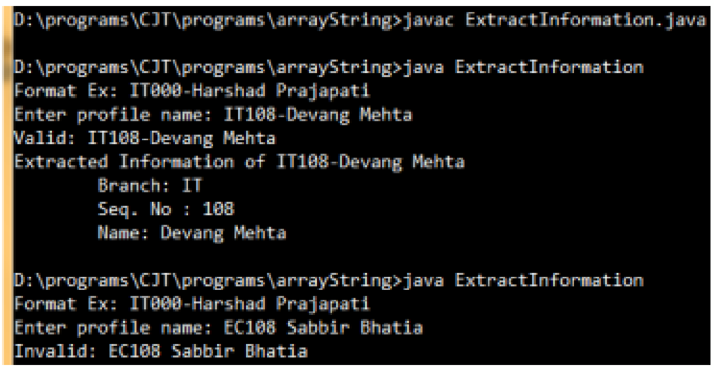
## Program: Extract Information, Slide - II

```java
         System.out.println("\tSeq. No : "+info[1]);
         System.out.println("\tName: "+info[2]);
      }
      else{
         printValidity(s);
      }

   }
   public static void printValidity(String s){
      if(isValid(s))
         System.out.println("Valid: "+s);
      else
         System.out.println("Invalid: "+s);
   }
   public static boolean isValid(String s){
      if(s.charAt(5)=='-')
         return true;
```

## Program: Extract Information, Slide - III

```java
      return false;
   }
   public static String[] extractInfo(String s){
      String[] result={"","",""};
      result[0]=s.substring(0,2);
      result[1]=s.substring(2,5);
      result[2]=s.substring(6);

      return result;
   }
}
```

## Program: Extract Information, Slide - IV



```
D:\programs\CJT\programs\arrayString>javac ExtractInformation.java

D:\programs\CJT\programs\arrayString>java ExtractInformation
Format Ex: IT000-Harshad Prajapati
Enter profile name: IT108-Devang Mehta
Valid: IT108-Devang Mehta
Extracted Information of IT108-Devang Mehta
        Branch: IT
        Seq. No : 108
        Name: Devang Mehta

D:\programs\CJT\programs\arrayString>java ExtractInformation
Format Ex: IT000-Harshad Prajapati
Enter profile name: EC108 Sabbir Bhatia
Invalid: EC108 Sabbir Bhatia
```

## String Methods

### Convert, Replace, Trim, and Split

```java
String toUpperCase();
String toLowerCase();
String trim();
```

- The toUpperCase() method returns a new string with all characters converted to uppercase.
- The toLowerCase() method returns a new string with all characters converted to lowercase.
- The trim() method removes leading and trailing blank characters.

## String Methods

### Convert, Replace, Trim, and Split

```java
String replace(char oldC, char newC);
String replaceFirst(String oldS, String newS);
String replaceAll(String oldS, String newS);
String[] split(String delimiter);
```

- The replace(oldC, newC); method returns a new string with all matching old characters replaced by new characters.
- The replaceFirst(oldS, newS); method returns a new string with the first matching substring (old) replaced by new substring.
- The replaceAll(oldS, newS); method returns a new string with all matching substrings (old) replaced by new substring.
- The split(delimiter); method returns an array of strings consisting of the substrings split by the specified delimiter.

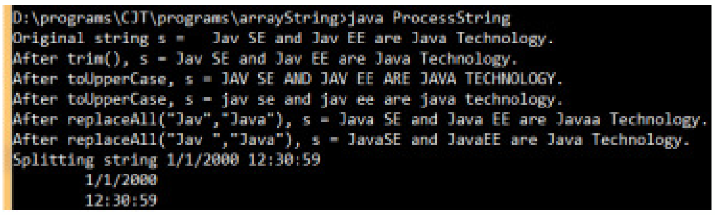
## Program: Processing a String, Slide - I

```java
class ProcessString{
    public static void main(String[] args){
        String s = "  Jav SE and Jav EE are Java Technology.  ";
        System.out.print("Original string s = ");
        System.out.println(s);
        s = s.trim();
        System.out.println("After trim(), s = "+s);
        System.out.println("After toUpperCase, s = "+s.toUpperCase());
        System.out.println("After toUpperCase, s = "+s.toLowerCase());
        System.out.println("After replaceAll(\"Jav\",\"Java\"), s = "+ s.replaceAll("Jav","Java"));
        System.out.println("After replaceAll(\"Jav \",\"Java\"), s = "+
```

## Program: Processing a String, Slide - II

```java
            s.replaceAll("Jav ","Java"));

        String dateTime = "1/1/2000 12:30:59";
        System.out.println("Splitting string "+dateTime);
        String[] parts = dateTime.split(" ");
        for(int i=0;i<parts.length;i++)
            System.out.println("\t"+parts[i]);
    }
}
```

## Program: Processing a String, Slide - III



```
D:\programs\CJT\programs\arrayString>java ProcessString
Original string s =    Jav SE and Jav EE are Java Technology.
After trim(), s = Jav SE and Jav EE are Java Technology.
After toUpperCase, s = JAV SE AND JAV EE ARE JAVA TECHNOLOGY.
After toUpperCase, s = jav se and jav ee are java technology.
After replaceAll("Jav","Java"), s = Java SE and Java EE are Javaa Technology.
After replaceAll("Jav ","Java"), s = JavaSE and JavaEE are Java Technology.
Splitting string 1/1/2000 12:30:59
        1/1/2000
        12:30:59
```

## String Methods

Knowing Index of String or Character

```java
int indexOf(char ch);
int indexOf(char ch, int fromIndex);
int indexOf(String s);
int indexOf(String s, int fromIndex);
```

- All these methods return -1, if match is not found.
- The indexOf(ch); returns the first occurrence of character ch in the invoking string.
- The indexOf(ch, fromIndex); returns the first occurrence of character ch after fromIndex, in the invoking string.
- The indexOf(s); returns the first occurrence of string s in the invoking string.
- The indexOf(s, fromIndex); returns the first occurrence of string s after fromIndex, in the invoking string.

## String Methods

Knowing Index of String or Character

```java
int lastIndexOf(char ch);
int lastIndexOf(char ch, int fromIndex);
int lastIndexOf(String s);
int lastIndexOf(String s, int fromIndex);
```

- All these methods return -1, if match is not found.
- The lastIndexOf(ch); returns the last occurrence of character ch in the invoking string.
- The lastIndexOf(ch, fromIndex); returns the last occurrence of character ch before fromIndex, in the invoking string.
- The lastIndexOf(s); returns the last occurrence of string s in the invoking string.
- The lastIndexOf(s, fromIndex); returns the last occurrence of string s before fromIndex, in the invoking string.

## String Methods

Match, Replace, and Split with Patterns

```java
boolean matches(String re);
String replaceFirst(String re, String s);
String replaceAll(String re, String s);
String[] split(String re);
```

- For comparing two strings, we use equals().
- However, matches() provides us to check equality using regular expression-RE (pattern).
- We can also pass regular expression (pattern) to replaceFirst(), replaceAll() and split() methods.

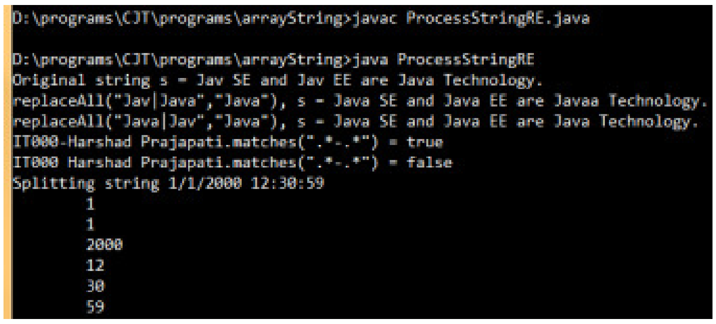
## Program: Processing a String with RE, Slide - I

```java
class ProcessStringRE{
   public static void main(String[] args){
      String s = "Jav SE and Jav EE are Java Technology.";
      String profileName1 = "IT000-Harshad Prajapati";
      String profileName2 = "IT000 Harshad Prajapati";
      String dateTime = "1/1/2000 12:30:59";


      System.out.print("Original string s = ");
      System.out.println(s);

      System.out.println("replaceAll(\"Jav|Java\", \"Java\"), s = "+s.replaceAll("Jav|Java","Java"));
```

## Program: Processing a String with RE, Slide - II

```java
      System.out.println("replaceAll(\"Java|Jav\", \"Java\"), s = "+s.replaceAll("Java|Jav","Java"));

      System.out.println(profileName1+".matches(\".*-.*\") = "+profileName1.matches(".*-.*"));
      System.out.println(profileName2+".matches(\".*-.*\") = "+profileName2.matches(".*-.*"));

      System.out.println("Splitting string "+dateTime);
      String[] parts = dateTime.split("[ /:]");
      for(int i=0;i<parts.length;i++)
         System.out.println("\t"+parts[i]);
   }
}
```

## Program: Processing a String with RE, Slide - III



```
D:\programs\CJT\programs\arrayString>javac ProcessStringRE.java

D:\programs\CJT\programs\arrayString>java ProcessStringRE
Original string s = Jav SE and Jav EE are Java Technology.
replaceAll("Jav|Java","Java"), s = Java SE and Java EE are Javaa Technology.
replaceAll("Java|Jav","Java"), s = Java SE and Java EE are Java Technology.
IT000-Harshad Prajapati.matches(".*-.*") = true
IT000 Harshad Prajapati.matches(".*-.*") = false
Splitting string 1/1/2000 12:30:59
        1
        1
        2000
        12
        30
        59
```

## Outline of Presentation

1. String class
2. StringBuffer class
3. StringTokenizer class
4. Command-Line Arguments

## StringBuffer and StringBuilder

- String object is immutable.
- StringBuffer and StringBuilder classes allow us to add, insert, or append new contents.
- StringBuffer and StringBuilder provide replacement for String class
- StringBuffer and StringBuilder are similar
  - StringBuffer is used if it is to be accessed among multiple threads. (Methods are synchronized)
  - StringBuilder is used if it is not to be accessed among multiple threads.
- Following constructors allow us to create StringBuffer object

```java
StringBuffer(); //empty with capacity 16
StringBuffer(int capacity); //empty with indicated capacity
StringBuffer(String s); //with capacity=length of string plus 16
```

## Methods of StringBuffer

- There are many forms of append() and insert() methods, i.e., for various primitive types, char array, and string.We show only widely used methods.

```java
StringBuffer append(String s);
StringBuffer insert(int offset, String s);
StringBuffer delete(int startIndex, int endIndex);
StringBuffer deleteCharAt(int index);
```

- The append() method appends given string s into the StringBuffer.
- The insert() method inserts the given string s at the specified offset in StringBuffer.
- The delete() method deletes characters from startIndex to endIndex-1.
- The deleteCharAt() method deletes character at the specified index.

## Methods of StringBuffer

```java
StringBuffer replace(int startIndex, int endIndex, String s);
StringBuffer reverse();
void setCharAt(int index, char ch);
```

- The replace() method replaces the characters from startIndex to endIndex-1 with the specified string s.
- The reverse() method reverses the characters of the StringBuffer.
- The setCharAt() method sets a new character at the specified index.

## Methods of StringBuffer

```java
String toString();
String substring(int startIndex);
String substring(int startIndex, int endIndex);
char charAt(int index);
int length();

int capacity();
void setLength(int newLength);
void trimToSize();
```

- The methods: toString(), substring(), charAt(), and length() are similar to that of String class.
- The capacity(); method returns the capacity of the StringBuffer.
- The setLength(); method sets new length of StringBuffer.
- The trimToSize(); method reduces storage size used for the StringBuffer.

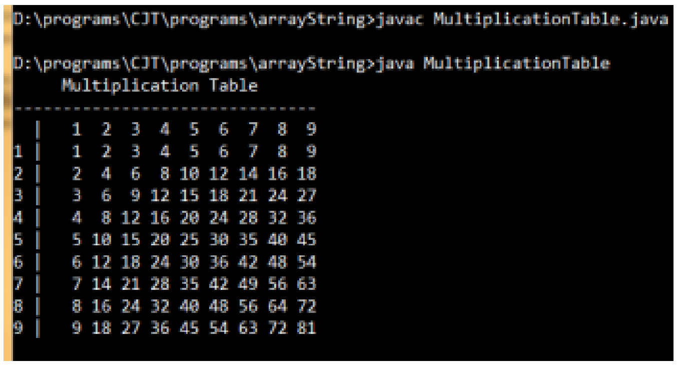
## Program: Mutiplication Table, Slide- I

```java
class MultiplicationTable{
   public static void main(String[] args){
      StringBuffer buffer = new StringBuffer();
      int product=0;
      buffer.append("      Multiplication Table"+'\n');
      buffer.append("--------------------");
      buffer.append("-----------"+'\n');

      buffer.append("   |  ");
      for(int j=1;j<=9;j++)
         buffer.append("   "+j);
      buffer.append('\n');

      for(int i=1;i<=9;i++){
         buffer.append(i+" |  ");
         for(int j=1;j<=9;j++){
            product = i*j;
```

## Program: Mutiplication Table, Slide- II

```java
            if(product<10)
               buffer.append("   "+product);
            else
               buffer.append("  "+product);
         }
         buffer.append("  "+'\n');
      }
      System.out.println(buffer);
   }
}
```

## Program: Mutiplication Table, Slide- III



```
D:\programs\CJT\programs\arrayString>javac MultiplicationTable.java

D:\programs\CJT\programs\arrayString>java MultiplicationTable
      Multiplication Table
-------------------------------
   |   1  2  3  4  5  6  7  8  9
1  |   1  2  3  4  5  6  7  8  9
2  |   2  4  6  8 10 12 14 16 18
3  |   3  6  9 12 15 18 21 24 27
4  |   4  8 12 16 20 24 28 32 36
5  |   5 10 15 20 25 30 35 40 45
6  |   6 12 18 24 30 36 42 48 54
7  |   7 14 21 28 35 42 49 56 63
8  |   8 16 24 32 40 48 56 64 72
9  |   9 18 27 36 45 54 63 72 81
```

## Outline of Presentation

1. String class
2. StringBuffer class
3. StringTokenizer class
4. Command-Line Arguments

## StringTokenizer class

- StringTokenizer class is available in java.util package.
- StringTokenizer breaks a string into pieces so that information contained in string can be retrieved and processed.
- For example, we want to separate out all words of a sentence.
- While constructing StringTokenizer, We can specify a set of delimiters, which break string into pieces.
- These pieces are called tokens.
- For example, for a string "Welcome to Java World", the tokens are Welcome, to, Java, and World.

## StringTokenizer class

- There are three forms of constructors:

```
StringTokenizer(String s);
StringTokenizer(String s, String delim);
StringTokenizer(String s, String delim,
   boolean returnTokens);
```

- The first constructor creates a string tokenizer for the give string s with delimiters as white space characters, '' \t\n\r'' (a space, tab, new line, and carriage return), and delimiters are not returned as tokens.
- The second form is similar to the first one, but it allows to specify delimiters.
- The third form is similar to the second one, but allows to specify whether tokens should be returned or not.

## Methods of StringTokenizer

```
boolean hasMoreTokens();
String nextToken();
String nextToken(String delim);
```

- The hasMoreTokens(); method returns true if there is any token left in the string. It is generally used in loop continue-condition.
- The nextToken(); method returns the next token in the string.
- The nextToken(delim); method returns the next token after resetting the delimiter to the specified delim.

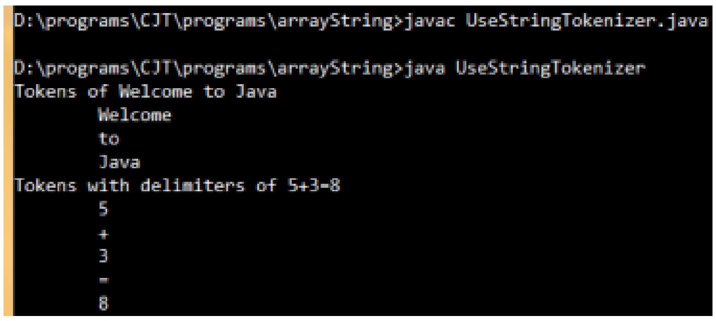
## Program: Use of StringTokenizer, Slide - I

```
import java.util.StringTokenizer;
class UseStringTokenizer{
   public static void main(String[] args){
      StringTokenizer st1=new
         StringTokenizer("Welcome to Java");
      System.out.println("Tokens of "+"Welcome
         to Java");
      while(st1.hasMoreTokens()){
         System.out.println("\t"+
            st1.nextToken());
      }
      StringTokenizer st2=new
         StringTokenizer("5+3=8", "+=", true);
      System.out.println("Tokens with
         delimiters of "+"5+3=8");
      while(st2.hasMoreTokens()){
         System.out.println("\t"+
            st2.nextToken());
```

## Program: Use of StringTokenizer, Slide - II

```
      }
   }
}
```

## Program: Use of StringTokenizer, Slide - III



```
D:\programs\CJT\programs\arrayString>javac UseStringTokenizer.java

D:\programs\CJT\programs\arrayString>java UseStringTokenizer
Tokens of Welcome to Java
        Welcome
        to
        Java
Tokens with delimiters of 5+3=8
        5
        +
        3
        =
        8
```

## Outline of Presentation

1 String class

2 StringBuffer class

3 StringTokenizer class

4 Command-Line Arguments

## Passing Command-Line Arguments

- Earlier we learned to supply input to our program using two different ways:
  1. Using Scanner class
  2. Using command-line arguments
- We provide command-line arguments when we run our program using java interpreter, for example
  - java Program arg0 arg1 arg2
- These command-line arguments (arg0 arg1 arg2) are placed into a String array and are passed to main() method.
  - main(String[ ] args)
- Inside main() method, these arguments can be accessed using args[0], args[1], args[2], ...
- Each word separated by white space is considered as one argument.
- If argument itself contains white spaces, then it should be enclosed in double quotes, e.g., "Good Morning"

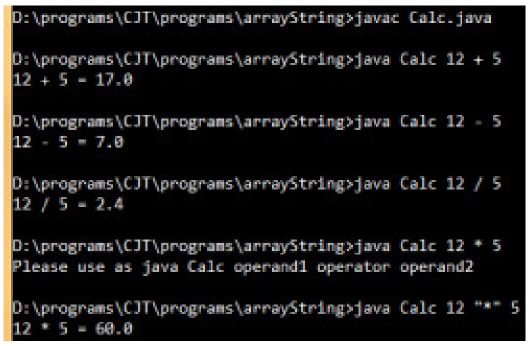
## Program: Command-line Calculator, Slide - I

```
class Calc{
   public static void main(String[] args){
      int no1,no2;
      double result=0;
      char operator;
      if(args.length!=3){
         System.out.println("Please use as
            java Calc operand1 operator
            operand2");
         System.exit(0);
      }
      no1=Integer.parseInt(args[0]);
      operator=args[1].charAt(0);
      no2=Integer.parseInt(args[2]);
      switch(operator){
         case '+':result=no1+no2;
            break;
         case '-':result=no1-no2;
```

## Program: Command-line Calculator, Slide - II

```
            break;
         case '*':result=no1*no2;
            break;
         case '/':result=(no1*1.0)/no2;
            break;
      }
      System.out.println(no1+" "+operator+"
         "+no2+" = "+result);
   }
}
```

## Program: Command-line Calculator, Slide - III



```
D:\programs\CJT\programs\arrayString>javac Calc.java

D:\programs\CJT\programs\arrayString>java Calc 12 + 5
12 + 5 = 17.0

D:\programs\CJT\programs\arrayString>java Calc 12 - 5
12 - 5 = 7.0

D:\programs\CJT\programs\arrayString>java Calc 12 / 5
12 / 5 = 2.4

D:\programs\CJT\programs\arrayString>java Calc 12 * 5
Please use as java Calc operand1 operator operand2

D:\programs\CJT\programs\arrayString>java Calc 12 "*" 5
12 * 5 = 60.0
```

## Summary of key terms

- String in Java, String class, Immutable string, interned string, string comparison, string ordering
- String methods
- StringBuffer and StringBuilder, Methods of StringBuffer
- StringTokenizer, methods of StringTokenizer
- Command-line arguments

## References

- An Introduction to Java Programming, Y. Daniel Liang, PHI
- An Introduction to Java Programming, Y. Daniel Liang, Eigth Edition, Prentice Hall