CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 3: String Handling, Regular Expression and Wrapper class

Chapter 1

Objective: After completing this lesson you will be able to : * Gain an idea regarding Java string class, mutable and immutable strings * Learn about StringBuffer and StringBuilder classes * Gain an introduction to string manipulation	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 60 minutes	**Practical Duration:** 60 minutes
Total Duration: 120 minutes	

Chapter 1

1.1 String Class

In Java, string class is a series or chain of characters. If one types in the word "Hello" within a Java program, the sequence created by the 5 letters of the word form a string. A string in Java is an immutable object, which makes it constant. Hence, a string that has been created cannot be changed.

To create a string in Java programmers can use these two methods –

- **String literal**

  String s = "Hello";

  To create string literal use double quotes " ". If the string already exists, the JVM returns a reference to the existing instance. If an earlier instance does not exist, a new one is created.

- **Using the *'new'* keyword**

  String s = new String ("Hello");

Example of a Java string –

```
public class Demo{
  public static void main(String args[]){
```

***string creation with literal.**

```
        String str = 'Hey';
        char arch[]={'h','i'};
        String str2 = new String(arch);
```

*** string creation with new keyword –**

```
        String str3 = new String('String Demoing');
        System.out.println(str);
        System.out.println(str2);
```

```
System.out.println(str3);
  }
}
```

Output:

Hey
Hi
String Demoing

1.2 Mutable and Immutable String

* **Immutable string –** String objects in Java are immutable by default. It means that they cannot be modified or changed. Adding, removing or inserting any character from an immutable string is not possible. java.lang.String is immutable. Java strings are immutable in nature as the language utilizes the concept of string literals.

Example of immutable string in Java –

```
package hello1;

public final class ImmutableDemo {

        private String name;

        ImmutableDemo (String name) {
                this.name = name;
        }

        public String getName() {
                return name;
        }
```

```
        public static void main(String[] args) {
                ImmutableDemo obj = new ImmutableDemo('hello');
                System.out.println(obj.getName());
                obj.setName('new hello');
                System.out.println(obj.getName());
        }
}
```

Output: hello

*** Mutable string –** Mutable string in Java enables the modification of characters within a string. Java strings are not mutable by default. But once a string is made mutable, programmers can add, remove or edit characters within it. To make a string mutable programmers can use the StringBuffer and StringBuilder classes.

Example of mutable string in Java –

```
package hello1;
public class MutableDemo {
        private String name;
        MutableClass(String name) {
                this.name = name;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public static void main(String[] args) {
                MutableDemo obj = new MutableDemo('hello1');
                System.out.println(obj.getName());
```

```
            Obj.setName('new hello1');
            System.out.println(obj.getName());
        }
}
```

Output

hello1

new hello 1

1.3 StringBuffer and StringBuilder Class

*** StringBuffer –** The Java StringBuffer class is used for creating a mutable string. This class is similar to the Java String class but mutable. Using this class helps programmers modify the characters of strings. A StringBuffer class cannot be accessed by multiple threads.

Constructors of StringBuffer

i) **StringBuffer( ):** An empty stringBuffer with a capacity of 16 is created.

ii) **StringBuffer(String str):** Creates an empty stringBuffer using the specified string.

iii) **StringBuffer(int capacity):** An empty stringBuffer with specified capacity as length is created.

StringBuffer Example –

```
public class Testing {
   public static void main(String args[]) {
      StringBuffer sBuffer = new StringBuffer('testing');
      sBuffer.append(' Strings Buffers');
      System.out.println(sBuffer);
```

```
  }
}
```

Output: testing Strings Buffers

*** StringBuilder –** The Java StringBuilder class is used to create a modifiable or mutable string. It is similar to the StringBuffer class but differs as it is non-synchronized.

Constructors of StringBuilder

i) StringBuilder(): Creates an empty stringBuilder with a capacity of 16.

ii) StringBuilder(String str): Creates a stringBuilder with a specific string.

iii) StringBuilder(int length): Creates an empty stringbuilder with specified capacity as length.

StringBuilder Example –

```
import java.lang.StringBuilder;
public class Program {
        public static void main(String[] args) {
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < 4; i++) {
           builder.append('xyz ');
         }
        String result = builder.toString();
```

```
        System.out.println(result);
    }
        }
```

Output: xyz xyz xyz xyz

*** Difference stringBuffer and stringBuilder –**

StringBuffer	stringBuilder
StringBuffer is a synchronized class i.e. stringBuffer methods cannot be called simultaneously by more than one thread. It is thus considered as thread safe.	StringBuilder is a non-synchronized class i.e. stringBuilder methods can be called simultaneously by multiple threads. Hence, it is not thread safe.
StringBuffer is slow and has lower efficiency than stringBuilder.	StringBuilder is fast and has greater efficiency than stringBuffer.

1.4 String manipulation – Java string manipulation refers to the procedure for modifying, splicing, pasting, analyzing and parsing strings. It is a mechanism of the Java platform. Through manipulation, the characters within a string can be edited or modified to achieve specific results. For example, the first character of a string in lowercase can be changed into title-case with manipulation.

The methods of string manipulation are –

- replace
- contains
- replaceAll
- indexOf
- substring
- Equals

- lastIndexOf
- startsWith
- endsWith
- EqualsIgnoreCase
- toLowerCase
- toUpperCase
- isEmpty
- Length
- split

Java string manipulation example:

```
import java.util.Scanner;

public class Main {

  public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);


 String s = sc.nextLine();
  System.out.println(display(s));

  }

  public static String display(String s) {

    StringBuffer sb = new StringBuffer();

    char a = s.charAt(0);

    char b = s.charAt(1);

    if (a != 'j' && b != 'b')

      sb.append(s.substring(2));
```

```
 else if (a == 'j' && b != 'b')

     sb.append('j').append(s.substring(2));


 else if (a != 'j' && b == 'b')
     sb.append(s.substring(1));

    else

     sb.append(s.substring(0));

    return sb.toString();

  }
}
```

Practical (60 minutes)

a) See the example programme for Java immutable string below. Write the same programme for the class ImmutableExample, to achieve object value 'Hi'.

```
package hello1;

public final class ImmutableDemo {

   private String name;

   ImmutableDemo (String name) {

      this.name = name;

   }

   public String getName() {

      return name;

   }
```

```
    public static void main(String[] args) {
        ImmutableDemo obj = new ImmutableDemo('hello');
        System.out.println(obj.getName());

     obj.setName('new hello');

   System.out.println(obj.getName());
    }
}
```

b) See the example programme for Java mutable string below. Write the same programme for the class MutableExample, to output the object values 'hello 2' and 'hello3'.

```
package hello1;
public class MutableDemo {
    private String name;
    MutableClass(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```

```
public static void main(String[] args) {

    MutableDemo obj = new MutableDemo('hello1');

    System.out.println(obj.getName());

    Obj.setName('new hello1');


  System.out.println(obj.getName());

}

}
```

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. What is a string class ?

a) chain of characters

b) list of characters

c) table of characters

d) None of the mentioned

2. A Java string is _____ object.

a) a mutable

b) an immutable

c) a designated

d) None of the mentioned

3. Immutable objects are _____.

a) constant

b) instant

c) dynamic

d) None of the mentioned

4. Can the new keyword be used to create a new string class ?

a) yes

b) no

c) sometimes

d) only literals can create strings

5. To create string literals use double _____.

a) quotes

b) brackets

c) commas

d) None of the mentioned

6. A Java string class is _____.

a) an object

b) a subject

c) an integer block

d) None of the mentioned

7. Are Java strings mutable ?

a) Yes

b) No

c) only by using StringBuilder and StringBuffer

d) None of the mentioned

8. A StringBuffer class cannot be accessed by multiple _____.

a) objects

b) classes

c) threads

d) subclasses

9. StringBuilder() creates an empty stringBuilder with a capacity of ____.

a) 15

b) 16

c) 32

d) 64

10. What is the process of splicing and manipulating strings called in Java?

a) string editing

b) string manipulation

c) string association

d) None of the mentioned