22IT402 - Software Testing

Topics

- Selenium setup
- Control Statements
- Arrays
- Strings and Functions

Selenium setup

Install Java Development Kit (JDK):

Selenium WebDriver supports multiple programming languages, but Java is one of the most commonly used languages for Selenium automation.

Install the latest version of JDK compatible with your operating system from the official Oracle website or using a package manager like Homebrew (for macOS) or apt (for Linux).

Set up Integrated Development Environment (IDE):

Choose an IDE for writing and managing your Selenium test scripts. Popular choices include Eclipse, IntelliJ IDEA, and Visual Studio Code.

Install the IDE of your choice and configure it according to your preferences.

Download Selenium WebDriver JAR files:

Download the Selenium WebDriver Java language bindings (JAR files) from the Selenium website or Maven Central Repository.

Alternatively, you can use a build management tool like Maven or Gradle to manage dependencies and download the Selenium WebDriver JAR files automatically.

Set up WebDriver executable:

Download the WebDriver executable (e.g., chromedriver for Chrome, geckodriver for Firefox) compatible with the web browser you intend to automate.

Ensure that the WebDriver executable is compatible with the browser version installed on your system.

Add the directory containing the WebDriver executable to the system's PATH environment variable to enable Selenium to locate and use it.

Create a new Java project:

Open your chosen IDE and create a new Java project for your Selenium automation testing.

Configure the project settings, such as the JDK version, project structure, and build path.

Add Selenium WebDriver JAR files to the project:

Add the downloaded Selenium WebDriver JAR files to the project's build path or dependencies.

In Eclipse, you can right-click on the project, select "Build Path" > "Configure Build Path," and then add the Selenium JAR files to the "Libraries" tab.

If you're using Maven or Gradle, add the Selenium dependencies to the project's pom.xml or build.gradle file, respectively, and let the build tool handle the dependency resolution.

Write and execute a sample test script:

Write a simple test script using Selenium WebDriver APIs to automate a basic interaction with a web page.

Instantiate a WebDriver object (e.g., ChromeDriver, FirefoxDriver) and use its methods to navigate to a web page, interact with web elements, and perform assertions.

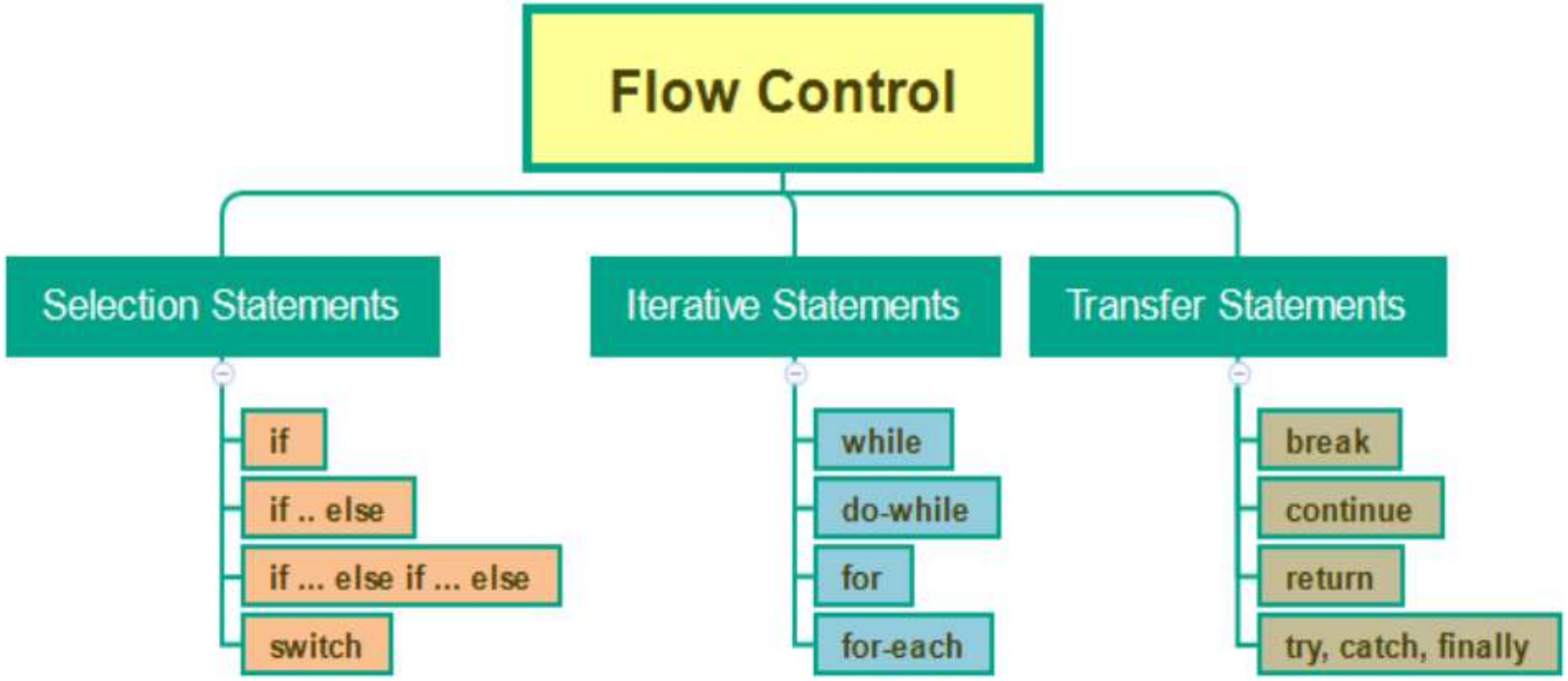
Execute the test script and verify that it runs successfully without errors.

(Optional) Install additional tools and plugins:

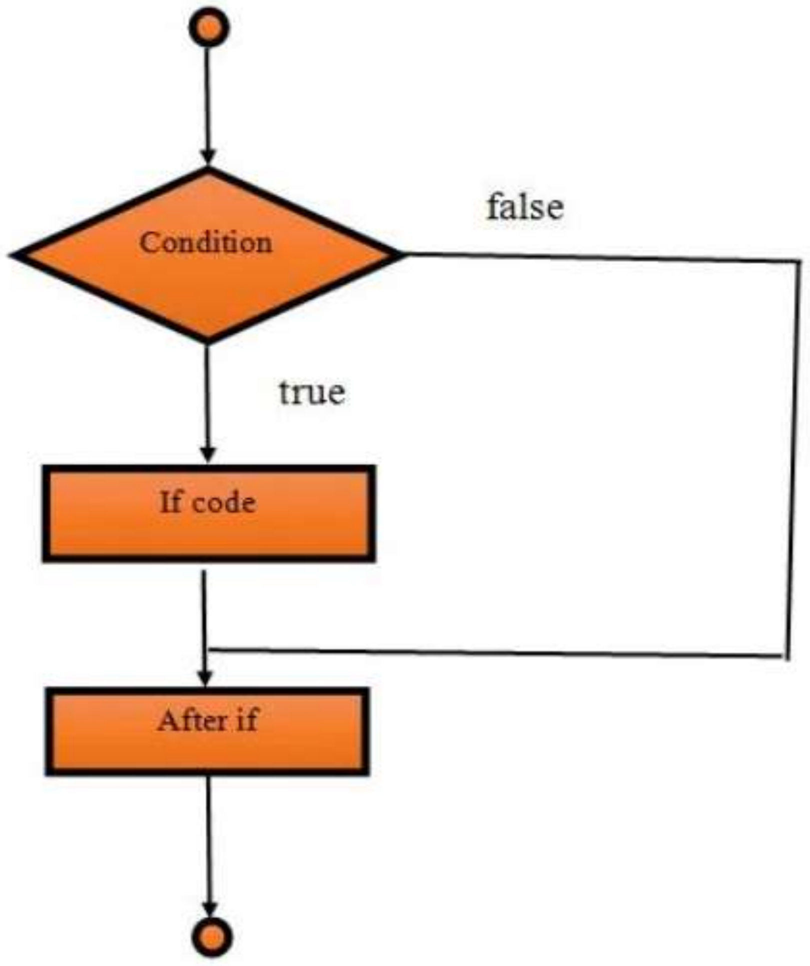
Depending on your requirements, you may need to install additional tools or plugins for enhancing your Selenium automation workflow.

For example, you can install testing frameworks like JUnit or TestNG, logging frameworks like Log4j, and reporting plugins like ExtentReports for better test management and reporting.

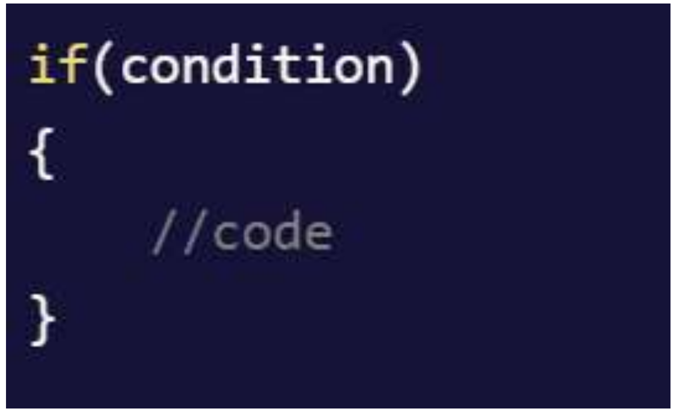
Control Statements



FLOW CHART OF IF STATEMENT:



SYNTAX:

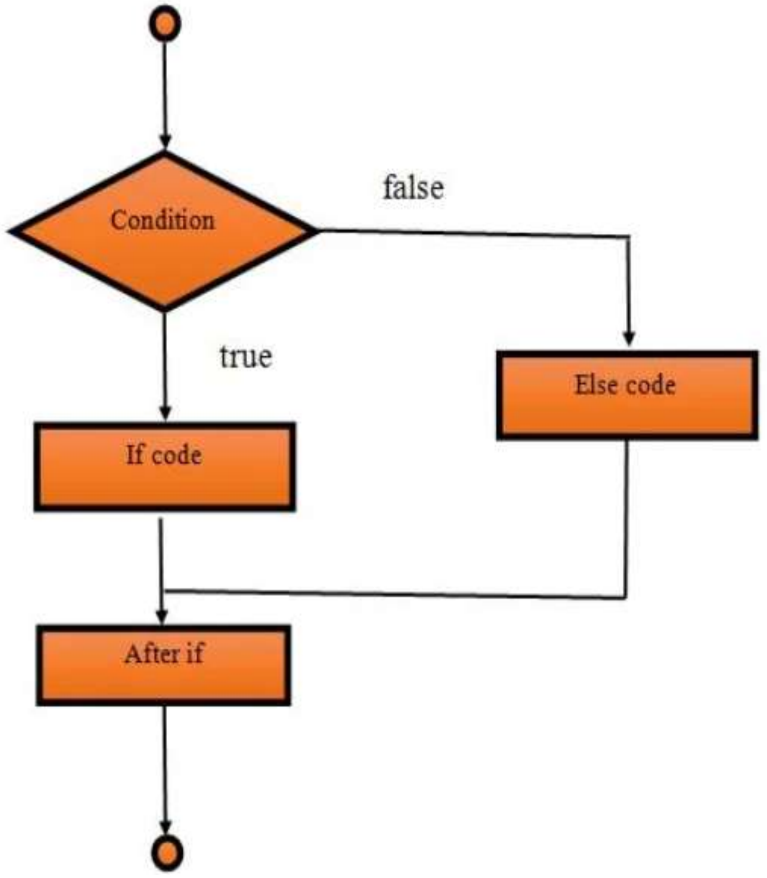


```
if(condition)
{
      //code
}
```

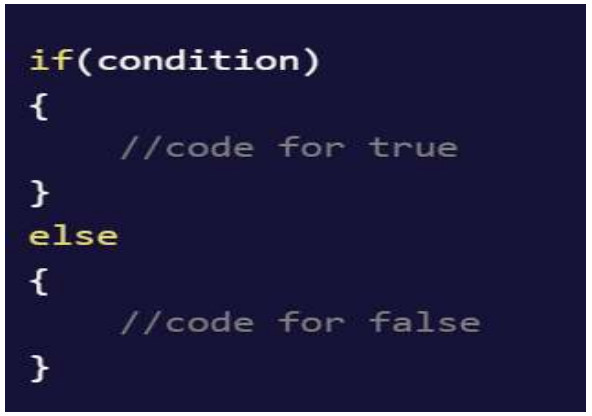
EXAMPLE:

```
public class IfDemo1 {
public static void main(String[] args)
            {
            int marks=70;
            if(marks > 65)
                        {
                        System.out.print("First division");
                        }
            }
}
```

FLOW CHART OF IF ELSE STATEMENT:



SYNTAX:

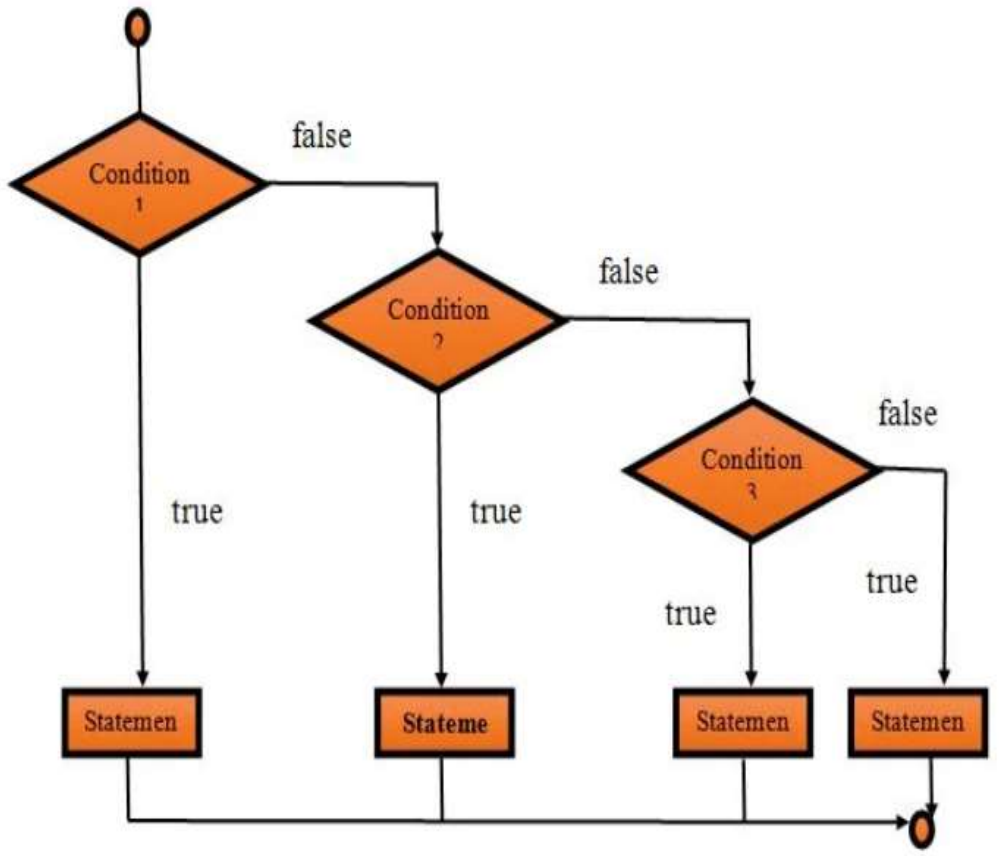


```
if(condition)
{
     //code for true
}
else
{
     //code for false
}
```

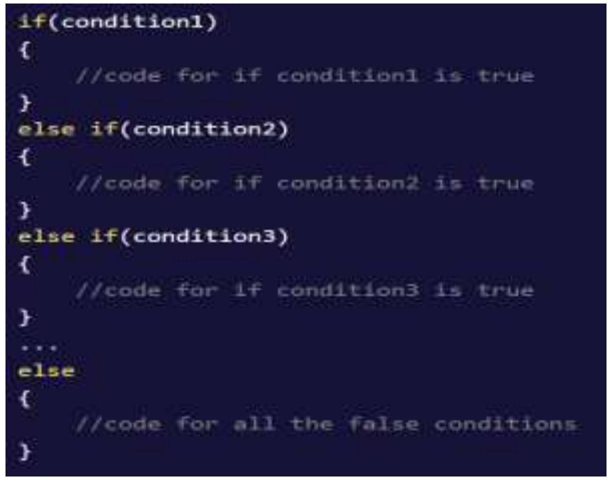
EXAMPLE:

```
public class IfElseDemo1 {
                public static void main(String[] args) {
            int marks=50;
            if(marks > 65)
            {
            System.out.print("First division");
            }
            else
            {
            System.out.print("Second division");
            }
            }
                }
```

FLOW CHART OF IF ELSE… IF ELSE STATEMENT:



SYNTAX:

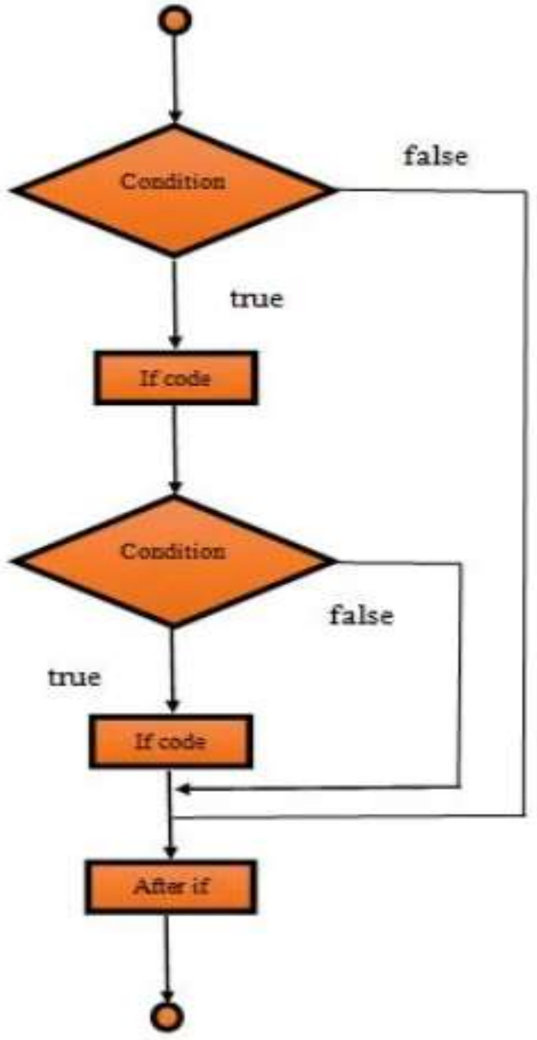


```
if(condition1)
{
    //code for if condition1 is true
}
else if(condition2)
{
    //code for if condition2 is true
}
else if(condition3)
{
    //code for if condition3 is true
}
...
else
{
    //code for all the false conditions
}
```

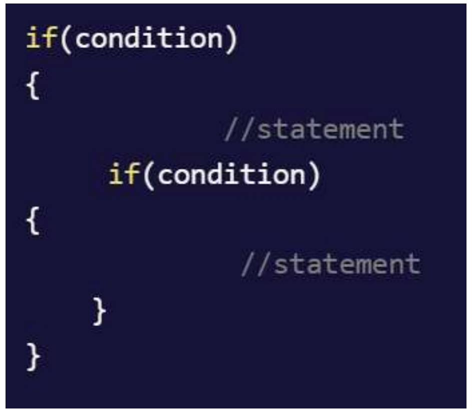
EXAMPLE:

```
        public class IfElseDemo1 {
        public static void main(String[] args) {
    int marks=50;
    if(marks > 65)
    {
    System.out.print("First division");
    }
    else
    {
    System.out.print("Second division");
    }
    }
        }
```

FLOW CHART OF SWITCH STATEMENT:



SYNTAX:



```
if(condition)
{
            //statement
     if(condition)
{
             //statement
    }
}
```

EXAMPLE:

```
public class NestedIfDemo1 {
public static void main(String[] args)
{
int age=25;
int weight=70;
if(age>=18)
{
if(weight>50)
{
System.out.println("You are eligible");
}
}
}
}
```

Arrays

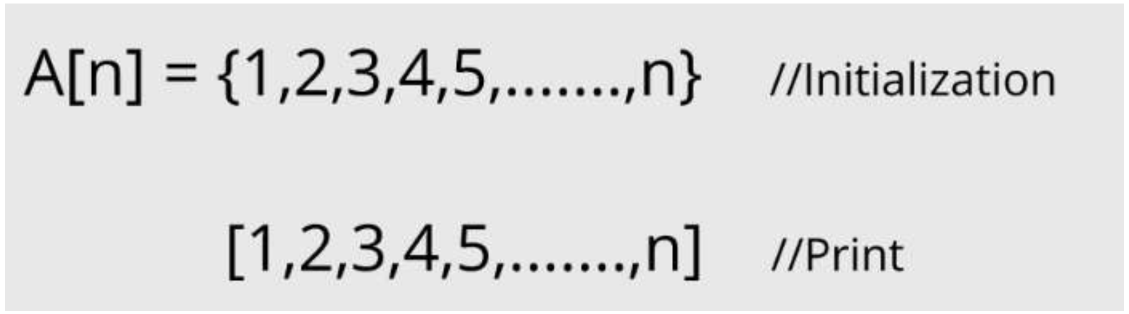
- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To declare an array, define the variable type with **square brackets**.

EXAMPLE:

```
public class Main {
public static void main(String[] args) {
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
}
}
```

OUTPUT: **Volvo**

In Java, an array is a fixed-size that stores elements of the same data type sequentially in memory. It allows efficient storage and retrieval of multiple values using indices. Arrays are declared with a specific type and size, and elements are accessed using their index position.



```
A[n] = {1,2,3,4,5,.......,n}    //Initialization

       [1,2,3,4,5,.......,n]    //Print
```

Types of Array

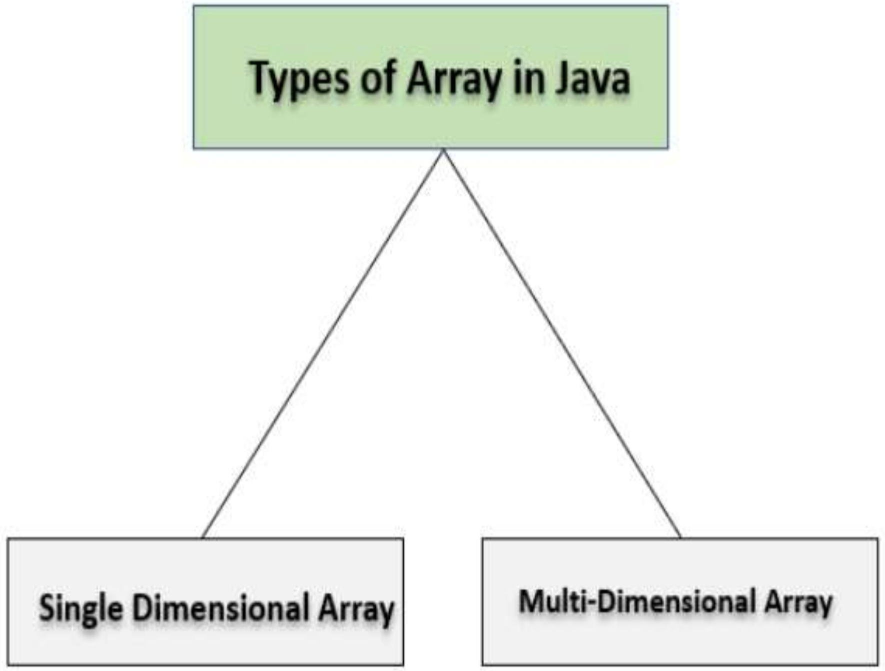
Arrays can be categorized into two main types:

1.Single-dimensional arrays:

Arrays that store elements of the same data type in a single row or column.

2. Multi-dimensional arrays:

Arrays that store elements in multiple rows and columns or in higher-dimensional structures, such as matrices or cubes. These can be two-dimensional, three-dimensional, or even more complex.



Strings

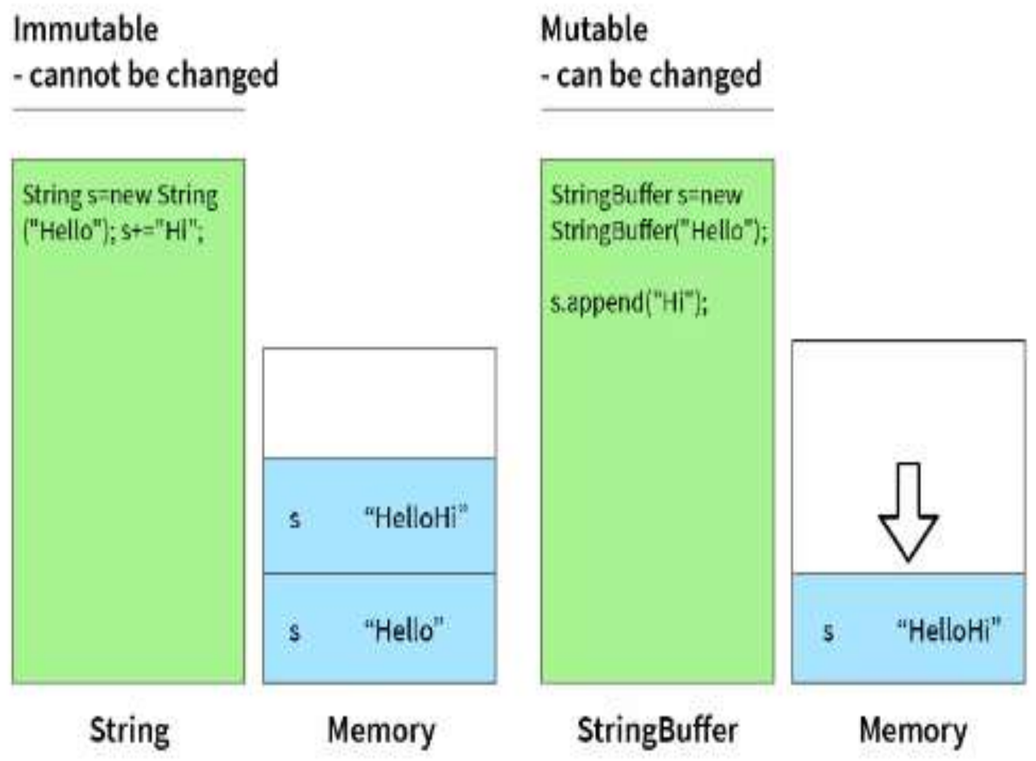
- A String variable contains a collection of characters surrounded by double quotes

EXAMPLE:

```
public class Main {
public static void main(String[]args) {
String greeting = "Hello";
System.out.println(greeting);
}
}
```

OUTPUT: Hello

- String is an **object** that represents sequence of characters. In Java, String is represented by String class which is located into java.lang package
- It is probably the most commonly used class in java library. In java, every string that we create is actually an object of type **String**. One important thing to notice about string object is that string objects are **immutable** that means once a string object is created it cannot be changed.



STRING FUNCTIONS:

String length

String charAt()

String charCodeAt()

String at()

String [ ]

String slice()

String substring()

String substr()

String toUpperCase()

String toLowerCase()

String concat()

String trim()

String trimStart()

String trimEnd()

String padStart()

String padEnd()

String repeat()

String replace()

String replaceAll()

String split()

EXAMPLE:

```
 public class StringFunctionsExample {
 public static void main(String[] args) {
// Example String
String originalString = "Hello, World!";

// 1. Length of the String
int length = originalString.length();
System.out.println("1. Length of the String: " + length);

// 2. Convert to Uppercase
String uppercaseString = originalString.toUpperCase();
System.out.println("2. Uppercase String: " + uppercaseString);

// 3. Convert to Lowercase
String lowercaseString = originalString.toLowerCase();
System.out.println("3. Lowercase String: " + lowercaseString);
```

```
// 4. Substring
String substring = originalString.substring(7); // Starting from index 7
System.out.println("4. Substring: " + substring);

// 5. Concatenation
String anotherString = " How are you?";
String concatenatedString = originalString.concat(anotherString);
System.out.println("5. Concatenated String: " + concatenatedString);

// 6. Replace
String replacedString = originalString.replace('o', '0');
System.out.println("6. Replaced String: " + replacedString);
```

```java
// 7. Index of
        int indexOfWorld = originalString.indexOf("World");
        System.out.println("7. Index of 'World': " + indexOfWorld);

        // 8. Equality check
        String compareString = "hello, world!";
        boolean isEqual = originalString.equals(compareString);
        System.out.println("8. Equality Check: " + isEqual);
    }
}
```

OUTPUT:

```
1. Length of the String: 13
2. Uppercase String: HELLO, WORLD!
3. Lowercase String: hello, world!
4. Substring: World!
5. Concatenated String: Hello, World! How are you?
6. Replaced String: Hell0, W0rld!
7. Index of 'World': 7
8. Equality Check: false
```