4.1. WAP in java to find out the address (IP) of any website.

**Algorithm:**

**Step 1:** Import the required Java libraries for networking and input handling.

**Step 2:** Define a class named **Test\_ip\_website**.

**Step 3:** Define the **main** method, which serves as the entry point for the program.

**Step 4:** Begin a try-catch block to handle potential exceptions that may occur during the program's execution.

**Step 5:** Inside the try block:

* Print a message prompting the user to enter the website name.
* Create a **Scanner** object to read user input.
* Read the website name entered by the user using **sc.nextLine()**.
* Use **InetAddress.getByName(name)** to obtain the IP address associated with the entered website name.
* Print the obtained IP address of the website.

**Step 6:** Add a catch block to handle the **UnknownHostException** exception. This exception is thrown when the website's address cannot be resolved.

**Step 7:** Inside the catch block, print an error message indicating that the website could not be found.

4.2. WAP in java to print the IP address of the local machine (in which the current program is running

**Algorithm:**

**Step 1:** Import the required Java library for networking.

**Step 2:** Define a public class.

**Step 3:** Define the **main** method, which serves as the entry point for the program.

**Step 4:** Begin a try-catch block to handle potential exceptions that may occur during the program's execution.

**Step 5:** Inside the try block, create an instance of the **InetAddress** class using the **getLocalHost()** method. This represents the localhost or the current computer's address.

**Step 6:** Print the obtained **InetAddress** object to the console, which will display the details about the localhost's address.

**Step 7:** Add a catch block to handle the **UnknownHostException** exception. This exception is thrown when the computer's address cannot be determined.

4.3. Check the version of IP address, whether it is version 4 or 6.

**Algorithm:**

**Step 1:** Import the necessary classes from the **java.net** package for working with IP addresses and sockets.

**Step 2:** Define a public class named **ipVersionCheck**.

**Step 3:** Define the **main** method, which is the entry point for the program.

**Step 4:** Start a try block to catch exceptions.

**Step 5:** Inside the try block:

* Resolve the IP address "2409:4089:1e91:3722:871:e41f:d2e2:5b45" using **InetAddress.getByName()**. Create an **InetAddress** object named **address** to store the result.

**Step 6:** Check if the resolved address (**address**) is an instance of **Inet6Address** using the **instanceof** operator. If it is, print "IPv6" to the console.

**Step 7:** If the resolved address is not an IPv6 address, check if it's an instance of **Inet4Address**. If it is, print "IPv4" to the console.

**Step 8:** Add a catch block to handle exceptions of type **Exception**. If any exception occurs during the execution, print "Some Error Occurred" to the console.

4.4. Print all the network interfaces (e.g., Ethernet, Bluetooth, wlan, etc.) of the local machine

**Algorithm:**

**Step 1:** Import the necessary classes from the **java.net** and **java.util** packages for working with network interfaces and collections.

**Step 2:** Define a public class named **InterfaceLister**.

**Step 3:** Inside the **InterfaceLister** class, define the **main** method. The method declaration includes **throws SocketException** to indicate that it may throw a **SocketException** if there is an issue with network interface enumeration.

**Step 4:** Retrieve an enumeration of all available network interfaces on the system using the **NetworkInterface.getNetworkInterfaces()** method. The **interfaces** variable holds this enumeration.

**Step 5:** Start a **while** loop that iterates through all the network interfaces in the **interfaces** enumeration.

**Step 6:** Within the loop, retrieve the next network interface in the enumeration using **interfaces.nextElement()** and store it in the **ni** variable.

**Step 7:** Print information about each network interface to the console using **System.out.println(ni)**. The **toString()** method of the **NetworkInterface** class is automatically called to provide a textual representation of the network interface. This representation includes its name, display name, and other relevant information.

**Step 8:** The loop continues until there are no more elements in the **interfaces** enumeration.