****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Fall, Year:2024), B.Sc. in CSE (Day)**

**Lab Report NO: 03**

**Course Title: Computer Networking Lab**

**Course Code: CSE-304 Section:221-D21**

**Lab Experiment Name: Implementation of socket programming using threading.**

**Student Details**

| **Name** | | **ID** |
| --- | --- | --- |
| **1.** | Masum Hossain | 221902164 |

**Lab Date : 29/09/2024**

**Submission Date : 15/11/2024**

**Course Teacher’s Name : Md. Saiful Islam Bhuiyan**

| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |
| --- |

**1. TITLE OF THE LAB REPORT EXPERIMENT**

Implementation of Iterative DNS Query for DNS Records

**2. OBJECTIVES**

This lab experiment involves creating a Java program to perform iterative DNS queries to retrieve IP addresses associated with given domain names. The aim is to understand and implement fundamental concepts of DNS resolution using iterative queries. The specific objectives of this lab experiment are:

* To understand the structure and functionality of DNS records.
* To implement an iterative approach to query DNS records for IP addresses.
* To analyze the efficiency of iterative queries compared to recursive queries.
* To demonstrate the retrieval process of DNS records through a simple Java application.

**3. ANALYSIS**

In this experiment, both iterative and recursive approaches were implemented to fetch the IP address corresponding to a domain name. The program maintains a list of DNS records and searches through these records according to the chosen method. The algorithm follows these steps:

**For Iterative Approach**

1. Create a list of DNS records containing domain names and their corresponding IP addresses.
2. Implement a method to search for an IP address based on a given domain name using iteration.
3. Return the IP address if found, or indicate that the domain was not found.

### **For Recursive Approach**

1. Create an array of DNS records containing domain names and their corresponding IP addresses.
2. Implement a method that searches for an IP address based on a given domain name using recursion.
3. Return the IP address if found, or proceed recursively until all records have been checked.

The main function allows the user to choose between the two methods and input a domain name to query.

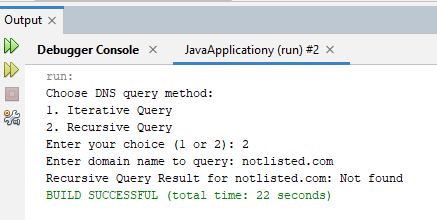
**4. IMPLEMENTATION**

The program is implemented in Java, using a simple class structure for DNS records along with two separate classes for iterative and recursive queries.

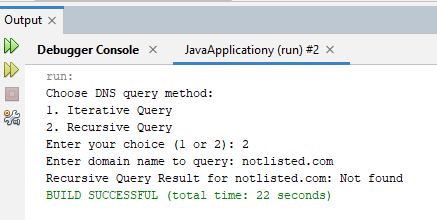
**Implemented Code**

| import java.util.ArrayList;  import java.util.List;  import java.util.Scanner;  class DNSRecord {  private String domain;  private String ipAddress;  public DNSRecord(String domain, String ipAddress) {  this.domain = domain;  this.ipAddress = ipAddress;  }  public String getDomain() {  return domain;  }  public String getIpAddress() {  return ipAddress;  }  }  class IterativeDNSQuery {  private List<DNSRecord> dnsRecords;  public IterativeDNSQuery() {  dnsRecords = new ArrayList<>();  dnsRecords.add(new DNSRecord("example.com", "93.184.216.34"));  dnsRecords.add(new DNSRecord("example.org", "2606:2800:220:1:248:1893:25c8:1946"));  dnsRecords.add(new DNSRecord("google.com", "172.217.14.206"));  dnsRecords.add(new DNSRecord("facebook.com", "157.240.22.35"));  }  public String findIpAddress(String domain) {  for (DNSRecord record : dnsRecords) {  if (record.getDomain().equals(domain)) {  return record.getIpAddress();  }  }  return "Not found";  }  }  class RecursiveDNSQuery {  private DNSRecord[] dnsRecords;  public RecursiveDNSQuery() {  dnsRecords = new DNSRecord[5];  dnsRecords[0] = new DNSRecord("example.com", "93.184.216.34");  dnsRecords[1] = new DNSRecord("example.org", "2606:2800:220:1:248:1893:25c8:1946");  dnsRecords[2] = new DNSRecord("google.com", "172.217.14.206");  dnsRecords[3] = new DNSRecord("linkedin.com", "192.0.2.1");  dnsRecords[4] = new DNSRecord("facebook.com", "157.240.22.35");  }  public String findIpAddress(String domain, int index) {  if (index >= dnsRecords.length) {  return "Not found";  }  if (dnsRecords[index].getDomain().equals(domain)) {  return dnsRecords[index].getIpAddress();  }  return findIpAddress(domain, index + 1); }}  public class DNSQueryProgram {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  IterativeDNSQuery iterativeQuery = new IterativeDNSQuery();  RecursiveDNSQuery recursiveQuery = new RecursiveDNSQuery();  System.out.println("Choose DNS query method:");  System.out.println("1. Iterative Query");  System.out.println("2. Recursive Query");  System.out.print("Enter your choice (1 or 2): ");  int choice = scanner.nextInt();    System.out.print("Enter domain name to query: ");  String domain = scanner.next();  if (choice == 1) {  System.out.println("Iterative Query Result for " + domain + ": " + iterativeQuery.findIpAddress(domain));  } else if (choice == 2) {  System.out.println("Recursive Query Result for " + domain + ": " + recursiveQuery.findIpAddress(domain, 0));  } else {  System.out.println("Invalid choice! Please enter 1 or 2.");  }  scanner.close(); }} |
| --- |

**OUTPUT**

The console output will display the result of the DNS query based on the user's input. For example, if the user queries "google.com," the output will look like this:

| **Fig 01: Showing the output for iterative Query listed on Array.** |
| --- |



| **Fig 02: Showing the output for Recursive Query not listed on Array.** |
| --- |

**6. ANALYSIS AND DISCUSSION**

The experiment successfully demonstrated the implementation of both iterative and recursive DNS queries within a Java program. By structuring DNS records and responding to user queries, we gained insights into the differences in implementation between the two methods.

The iterative approach proved straightforward and efficient for small datasets, while the recursive approach showcased the elegance of recursive function calls but may add overhead due to multiple stack frames. In practical applications, iterative queries often outperform recursive queries, especially as the size of DNS records grows.The process of developing this application reinforced the understanding of how DNS resolution works and the programming concepts of lists.