**GURU TEGH BAHADUR INSTITUTE OF TECHNOLOGY**

# ADVANCED JAVA PROGRAMMING PRACTICAL FILE

**SUBJECT CODE: CIE-306T**

**SEMESTER – VI**

|  |  |
|---|---|
| **Submitted to,** | **Submitted by,** |
| **Dr. Amandeep Kaur** | **Name: Abhay Raj** |
|  | **Enrolment No: 00976803122** |
|  | **Branch: IT – 3** |
|  | **Batch: 2022-2026** |

# Index

## Program – 1

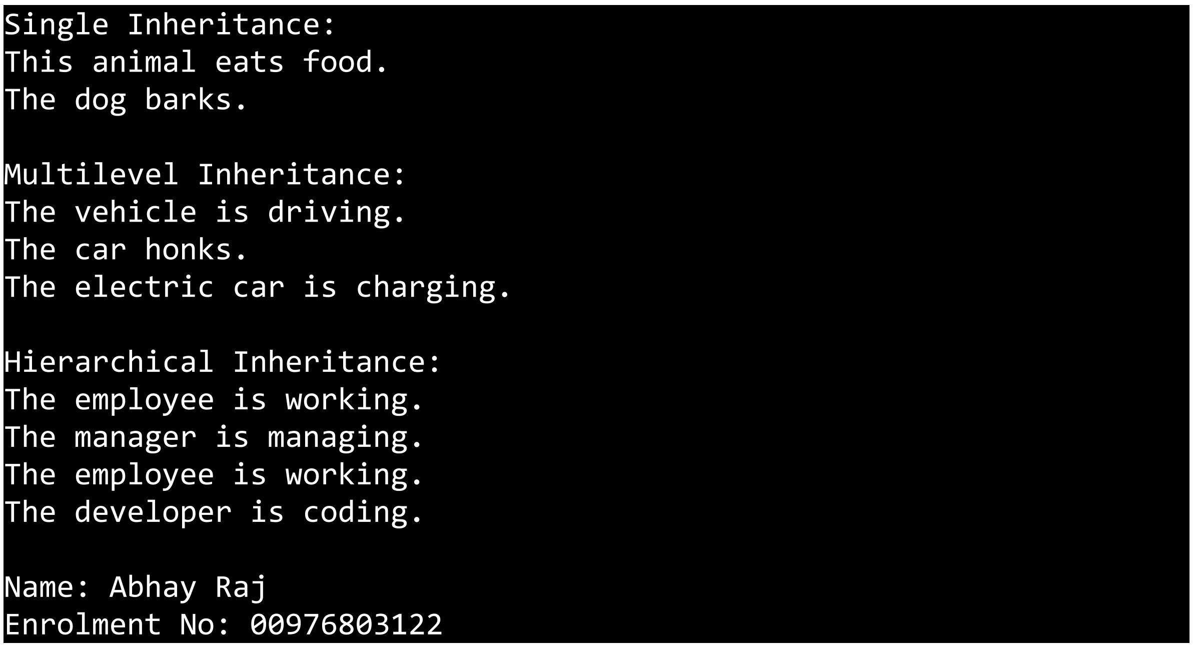
Aim: Write a program to demonstrate different types of inheritance.

Code:

```
class Animal {
  void eat() {
    System.out.println("This animal eats food.");
  }
}

class Dog extends Animal {
  void bark() {
    System.out.println("The dog barks.");
  }
}

class Vehicle {
  void drive() {
    System.out.println("The vehicle is driving.");
  }
}

class Car extends Vehicle {
  void honk() {
    System.out.println("The car honks.");
  }
}

class ElectricCar extends Car {
  void charge() {
    System.out.println("The electric car is charging.");
  }
}

class Employee {
  void work() {
    System.out.println("The employee is working.");
  }
}

class Manager extends Employee {
  void manage() {
    System.out.println("The manager is managing.");
  }
}

class Developer extends Employee {
  void code() {
    System.out.println("The developer is coding.");
```

```java
  }
}
public class InheritanceDemo {
  public static void main(String[] args) {
    System.out.println("Single Inheritance:");
    Dog dog = new Dog();
    dog.eat();
    dog.bark();
    System.out.println();

    System.out.println("Multilevel Inheritance:");
    ElectricCar electricCar = new ElectricCar();
    electricCar.drive();
    electricCar.honk();
    electricCar.charge();
    System.out.println();

    System.out.println("Hierarchical Inheritance:");
    Manager manager = new Manager();
    manager.work();
    manager.manage();

    Developer developer = new Developer();
    developer.work();
    developer.code();
    System.out.println();
    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
  }
}
```

Output:



```
Single Inheritance:
This animal eats food.
The dog barks.

Multilevel Inheritance:
The vehicle is driving.
The car honks.
The electric car is charging.

Hierarchical Inheritance:
The employee is working.
The manager is managing.
The employee is working.
The developer is coding.

Name: Abhay Raj
Enrolment No: 00976803122
```

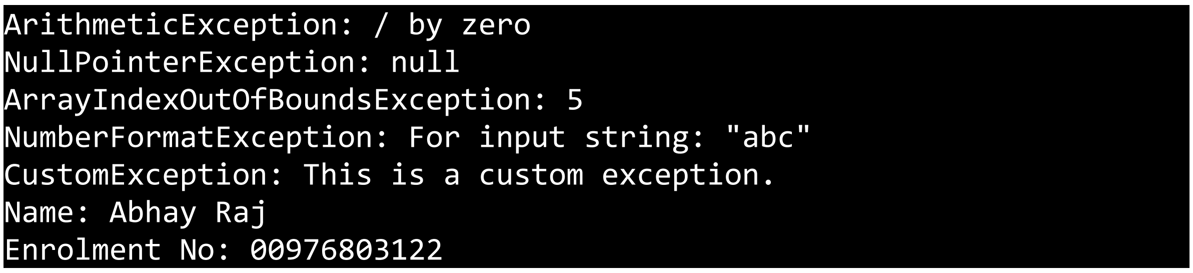
## Program – 2

Aim: Write a program to show different types of exceptions.

Code:

```java
class CustomException extends Exception {
  public CustomException(String message) {
    super(message);
  }
}

public class ExceptionDemo {
  public static void main(String[] args) {
    try {
      int result = 10 / 0;
    } catch (ArithmeticException e) {
      System.out.println("ArithmeticException: " + e.getMessage());
    }

    try {
      String str = null;
      System.out.println(str.length());
    } catch (NullPointerException e) {
      System.out.println("NullPointerException: " + e.getMessage());
    }

    try {
      int[] arr = new int[2];
      arr[5] = 10;
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("ArrayIndexOutOfBoundsException: " +
e.getMessage());
    }

    try {
      String str = "abc";
      int num = Integer.parseInt(str);
    } catch (NumberFormatException e) {
      System.out.println("NumberFormatException: " + e.getMessage());
    }

    try {
      throw new CustomException("This is a custom exception.");
    } catch (CustomException e) {
      System.out.println("CustomException: " + e.getMessage());
    }
    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
  }
}
```

Output:



```
ArithmeticException: / by zero
NullPointerException: null
ArrayIndexOutOfBoundsException: 5
NumberFormatException: For input string: "abc"
CustomException: This is a custom exception.
Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 3

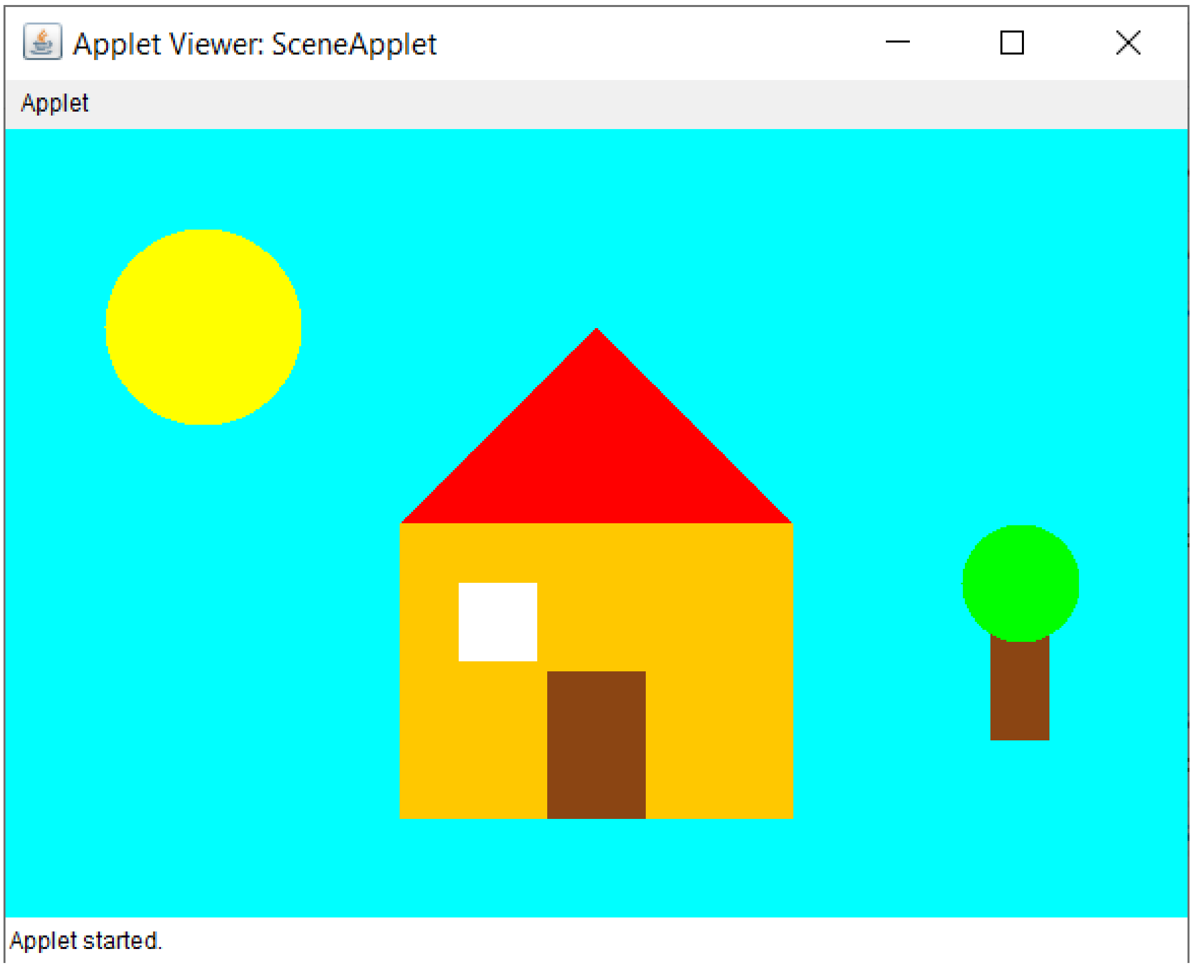
Aim: Write a program to create a scene using graphics class of Java applet.

Code:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;

public class SceneApplet extends Applet {
  public void paint(Graphics g) {

    setBackground(Color.cyan);

    g.setColor(Color.yellow);
    g.fillOval(50, 50, 100, 100);


    g.setColor(Color.orange);
    g.fillRect(200, 200, 200, 150);

    g.setColor(Color.red);
    int[] xPoints = {200, 400, 300};
    int[] yPoints = {200, 200, 100};
    g.fillPolygon(xPoints, yPoints, 3);


    g.setColor(new Color(139, 69, 19));
    g.fillRect(275, 275, 50, 75);


    g.setColor(Color.white);
    g.fillRect(230, 230, 40, 40);


    g.setColor(new Color(139, 69, 19));
    g.fillRect(500, 250, 30, 60);


    g.setColor(Color.green);
    g.fillOval(485, 200, 60, 60);
    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
  }
}
/*
<applet code="SceneApplet" width=600 height=400>
</applet>
*/
```

Output:

```
Name: Abhay Raj
Enrolment No: 00976803122
```



## Program – 4

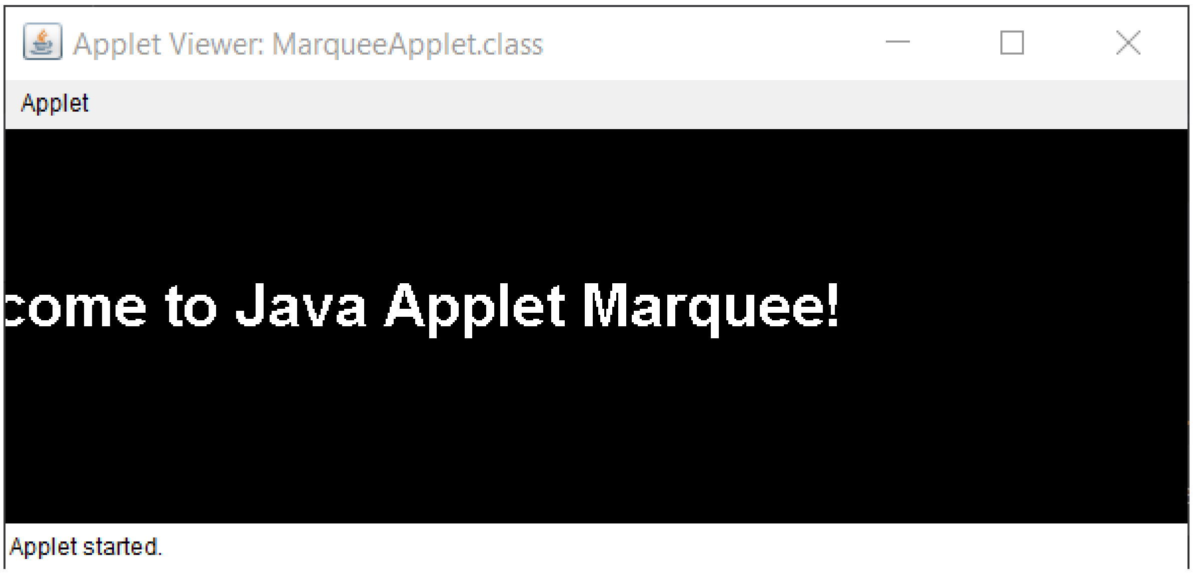
Aim: Write a program to create a Marquee using Java applet.

Code:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

public class MarqueeApplet extends Applet implements Runnable {
  private String message = "Welcome to Java Applet Marquee!";
  private int xPosition = 600; // Start position of the text
  private Thread marqueeThread;

  @Override
  public void init() {
    setBackground(Color.black);
    setForeground(Color.white);
    setFont(new Font("Arial", Font.BOLD, 30));

    // Start the marquee thread
    marqueeThread = new Thread(this);
    marqueeThread.start();
  }

  @Override
  public void run() {
    while (true) {
      // Move the text to the left
      xPosition -= 2;

      // If the text has completely moved off the screen, reset its position
      if (xPosition < -getFontMetrics(getFont()).stringWidth(message)) {
        xPosition = getWidth();
      }

      // Repaint the applet to update the position of the text
      repaint();

      try {
        // Pause the thread for 50 milliseconds to control the speed of the
scrolling
        Thread.sleep(50);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  @Override
  public void paint(Graphics g) {
    // Draw the message at the current x position
    g.drawString(message, xPosition, 100);
  }

  @Override
  public void stop() {
    // Stop the thread when the applet is stopped
    marqueeThread = null;
  }
}
// <html>
// <body>
//     <applet code="MarqueeApplet.class" width="600" height="200">
//     </applet>
// </body>
// </html>
```

Output:

```
Name: Abhay Raj
Enrolment No: 00976803122
```



```
come to Java Applet Marquee!
```

Applet started.

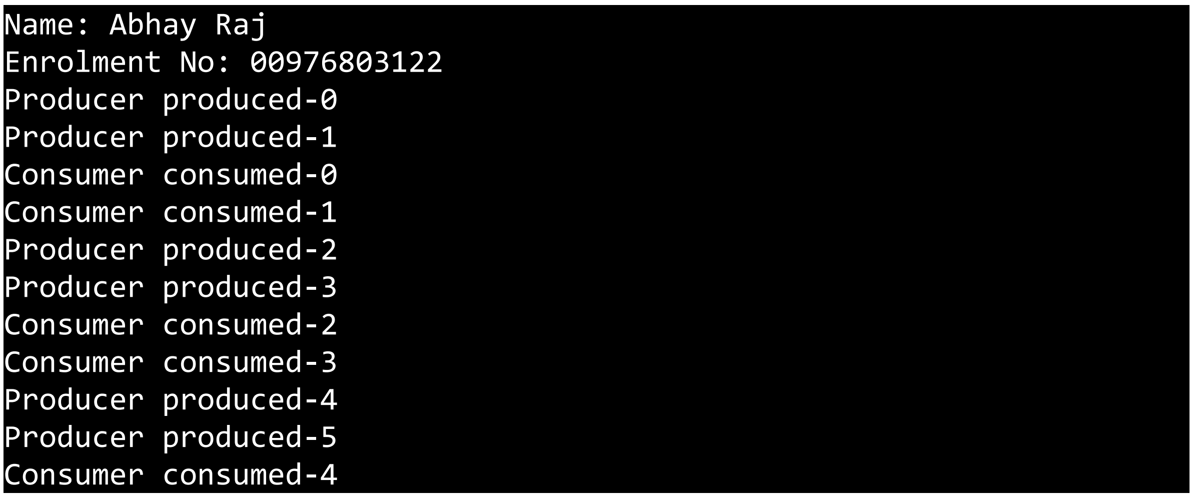
## Program – 5

Aim: Design a java program to implement producer consumer problem.

Code:

```
import java.util.LinkedList;

public class Threadexample {
  public static void main(String[] args)
    throws InterruptedException
  {
    final PC pc = new PC();
    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    Thread t1 = new Thread(new Runnable() {
      @Override
      public void run()
      {
        try {
          pc.produce();
        }
        catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    });

    Thread t2 = new Thread(new Runnable() {
      @Override
      public void run()
      {
        try {
          pc.consume();
        }
        catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    });

    t1.start();
    t2.start();


    t1.join();
    t2.join();

  }

  public static class PC {
    LinkedList<Integer> list = new LinkedList<>();
    int capacity = 2;

    public void produce() throws InterruptedException
    {
     int value = 0;
     while (true) {
      synchronized (this)
      {

       while (list.size() == capacity)
        wait();

       System.out.println("Producer produced-"
           + value);
       list.add(value++);

       notify();


       Thread.sleep(1000);
      }
     }
    }

    // Function called by consumer thread
    public void consume() throws InterruptedException
    {
     while (true) {
      synchronized (this)
      {
       // consumer thread waits while list
       // is empty
       while (list.size() == 0)
        wait();

       int val = list.removeFirst();

       System.out.println("Consumer consumed-"
           + val);

       notify();

       Thread.sleep(1000);
      }
     }
    }
   }
  }
```

Output:


```
Name: Abhay Raj
Enrolment No: 00976803122
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
Consumer consumed-3
Producer produced-4
Producer produced-5
Consumer consumed-4
```

## Program – 6

Aim: Design a java program to implement reader writer problem.

Code:

```
class SharedResource {
  private int data = 0;
  private int readers = 0;

  public synchronized void read() throws InterruptedException {
    while (readers == -1) {
      wait();
    }
    readers++;
    System.out.println("Reader reads: " + data);
    readers--;
    if (readers == 0) {
      notifyAll();
    }
  }

  public synchronized void write(int newData) throws InterruptedException {
    while (readers > 0) {
      wait();
    }
    readers = -1;
    data = newData;
    System.out.println("Writer writes: " + data);
    readers = 0;
    notifyAll();
  }
}

class Reader extends Thread {
  private SharedResource resource;

  public Reader(SharedResource resource) {
    this.resource = resource;
  }

  @Override
  public void run() {
    try {
      resource.read();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

```java
class Writer extends Thread {
  private SharedResource resource;

  public Writer(SharedResource resource) {
    this.resource = resource;
  }

  @Override
  public void run() {
    try {
      resource.write(100);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}

public class ReaderWriterProblem {
  public static void main(String[] args) {
    SharedResource resource = new SharedResource();
    Reader reader1 = new Reader(resource);
    Reader reader2 = new Reader(resource);
    Writer writer1 = new Writer(resource);

    reader1.start();
    reader2.start();
    writer1.start();

    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
  }
}
```

Output:

```
Name: Abhay Raj
Enrolment No: 00976803122
Reader reads: 0
Reader reads: 0
Writer writes: 100
```

## Program – 7

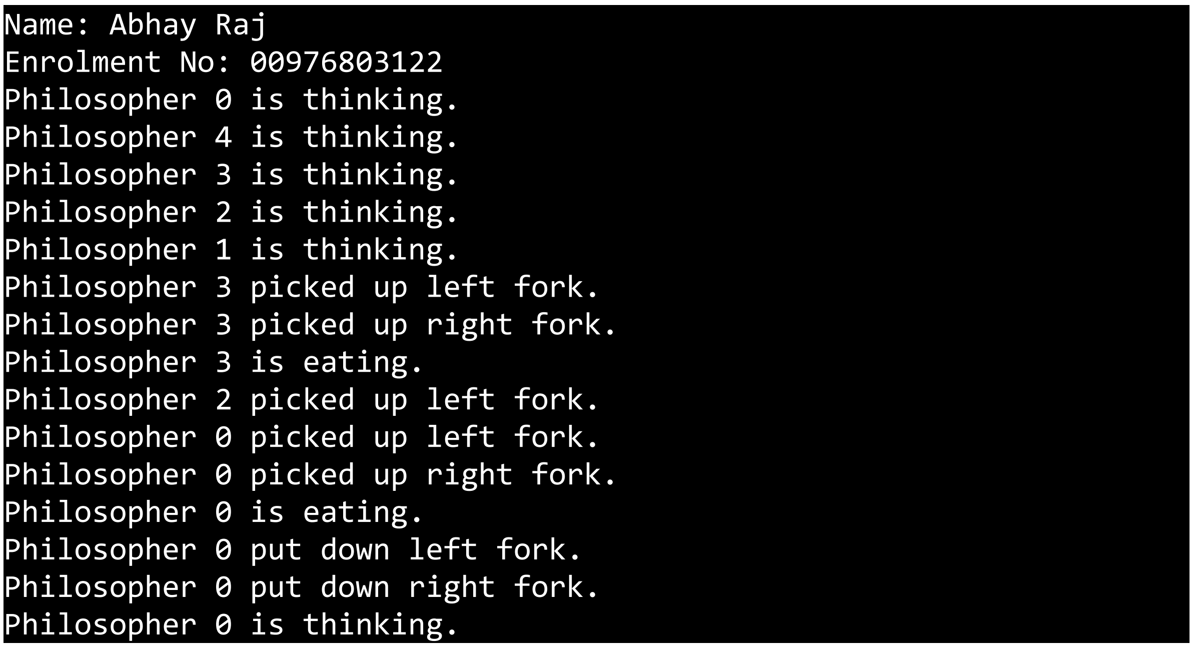
Aim: Design a java program to implement dining philosophers problem.

Code:

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class Philosopher extends Thread {
  private int id;
  private Lock leftFork;
  private Lock rightFork;

  public Philosopher(int id, Lock leftFork, Lock rightFork) {
    this.id = id;
    this.leftFork = leftFork;
    this.rightFork = rightFork;
  }

  public void run() {
    try {
      while (true) {
        think();
        pickUpForks();
        eat();
        putDownForks();
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }

  private void think() throws InterruptedException {
    System.out.println("Philosopher " + id + " is thinking.");
    Thread.sleep((long) (Math.random() * 1000));
  }

  private void pickUpForks() throws InterruptedException {
    leftFork.lock();
    System.out.println("Philosopher " + id + " picked up left fork.");
    rightFork.lock();
    System.out.println("Philosopher " + id + " picked up right fork.");
  }

  private void eat() throws InterruptedException {
    System.out.println("Philosopher " + id + " is eating.");
    Thread.sleep((long) (Math.random() * 1000));
  }

  private void putDownForks() {
```

```java
    leftFork.unlock();
    System.out.println("Philosopher " + id + " put down left fork.");
    rightFork.unlock();
    System.out.println("Philosopher " + id + " put down right fork.");
  }
}

public class DiningPhilosophers {
  public static void main(String[] args) {
    int n = 5;
    Lock[] forks = new Lock[n];

    System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");

    for (int i = 0; i < n; i++) {
      forks[i] = new ReentrantLock();
    }

    Philosopher[] philosophers = new Philosopher[n];
    for (int i = 0; i < n; i++) {
      philosophers[i] = new Philosopher(i, forks[i], forks[(i + 1) % n]);
      philosophers[i].start();
    }
  }
}
```

Output:



```
Name: Abhay Raj
Enrolment No: 00976803122
Philosopher 0 is thinking.
Philosopher 4 is thinking.
Philosopher 3 is thinking.
Philosopher 2 is thinking.
Philosopher 1 is thinking.
Philosopher 3 picked up left fork.
Philosopher 3 picked up right fork.
Philosopher 3 is eating.
Philosopher 2 picked up left fork.
Philosopher 0 picked up left fork.
Philosopher 0 picked up right fork.
Philosopher 0 is eating.
Philosopher 0 put down left fork.
Philosopher 0 put down right fork.
Philosopher 0 is thinking.
```

## Program – 8

Aim: Write a program for a client to server communication.

Code (Client.java):

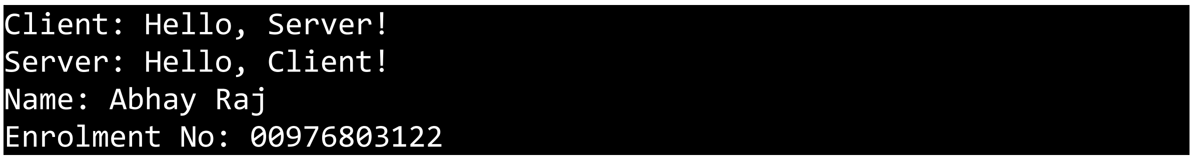
```
import java.io.*;
import java.net.*;

public class Client {
  public static void main(String[] args) {
    try (Socket socket = new Socket("localhost", 1234)) {
      PrintWriter output = new PrintWriter(socket.getOutputStream(), true);
      BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

      String message = "Hello, Server!";
      output.println(message);
      System.out.println("Client: " + message);

      String serverResponse = input.readLine();
      System.out.println(serverResponse);

      socket.close();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Code (Server.java)

```
import java.io.*;
import java.net.*;

public class Server {
  public static void main(String[] args) {
    try (ServerSocket serverSocket = new ServerSocket(1234)) {
      System.out.println("Server is waiting for client connection...");
      Socket clientSocket = serverSocket.accept();
      System.out.println("Client connected!");

      BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      PrintWriter output = new PrintWriter(clientSocket.getOutputStream(),
true);

      String clientMessage = input.readLine();
      System.out.println("Client: " + clientMessage);
```

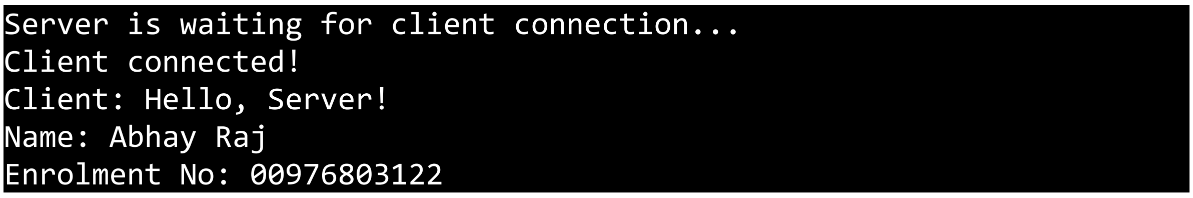
```java
      String response = "Server: Hello, Client!";
      output.println(response);

      clientSocket.close();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Output (Client.java):



```
Client: Hello, Server!
Server: Hello, Client!
Name: Abhay Raj
Enrolment No: 00976803122
```

Output (Server.java):



```
Server is waiting for client connection...
Client connected!
Client: Hello, Server!
Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 9

Aim: Write a program to implement a server to client communication.

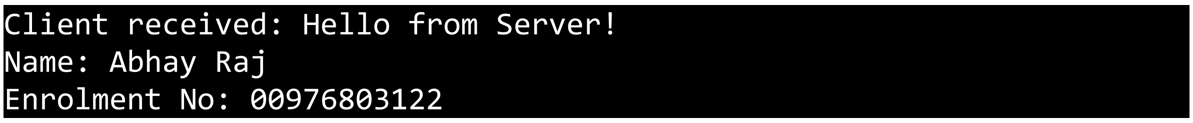
Code (Client.java):

```
import java.io.*;
import java.net.*;

public class Client {
  public static void main(String[] args) {
    try (Socket socket = new Socket("localhost", 1234)) {
      BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

      String serverMessage = input.readLine();
      System.out.println("Client received: " + serverMessage);

      socket.close();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Code (Server.java):

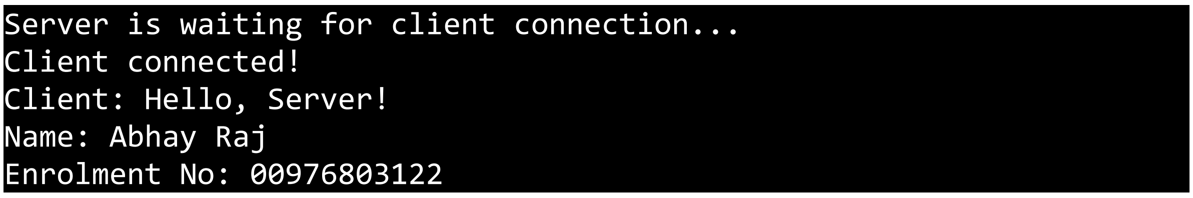
```
import java.io.*;
import java.net.*;

public class Server {
  public static void main(String[] args) {
    try (ServerSocket serverSocket = new ServerSocket(1234)) {
      System.out.println("Server is waiting for client connection...");
      Socket clientSocket = serverSocket.accept();
      System.out.println("Client connected!");

      PrintWriter output = new PrintWriter(clientSocket.getOutputStream(),
true);
      String serverMessage = "Hello from Server!";
      output.println(serverMessage);
      System.out.println("Server sent: " + serverMessage);
      clientSocket.close();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Output (Client.java):



```
Client received: Hello from Server!
Name: Abhay Raj
Enrolment No: 00976803122
```

Output (Server.java):



```
Server is waiting for client connection...
Client connected!
Client: Hello, Server!
Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 10

Aim: Write a program to implement a bidirectional communication.

Code (Client.java):

```
import java.io.*;
import java.net.*;

public class Client {
  public static void main(String[] args) {
    try (Socket socket = new Socket("localhost", 1234)) {
      BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
      PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

      String clientMessage = "Hello from Client!";
      output.println(clientMessage);
      System.out.println("Sent to Server: " + clientMessage);

      String serverResponse = input.readLine();
      System.out.println("Received from Server: " + serverResponse);
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
      socket.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Code (Server.java):

```
import java.io.*;
import java.net.*;

public class Server {
  public static void main(String[] args) {
    try (ServerSocket serverSocket = new ServerSocket(1234)) {
      System.out.println("Server is waiting for client connection...");
      Socket clientSocket = serverSocket.accept();
      System.out.println("Client connected!");

      BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      PrintWriter output = new PrintWriter(clientSocket.getOutputStream(),
true);

      String clientMessage = input.readLine();
      System.out.println("Received from Client: " + clientMessage);

      String serverResponse = "Hello from Server!";
      output.println(serverResponse);
      System.out.println("Sent to Client: " + serverResponse);
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
      clientSocket.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Output (Client.java):

```
Sent to Server: Hello from Client!
Received from Server: Hello from Server!
Name: Abhay Raj
Enrolment No: 00976803122
```

Output (Server.java):

```
Server is waiting for client connection...
Client connected!
Client: Hello, Server!
Name: Abhay Raj
Enrolment No: 00976803122
```

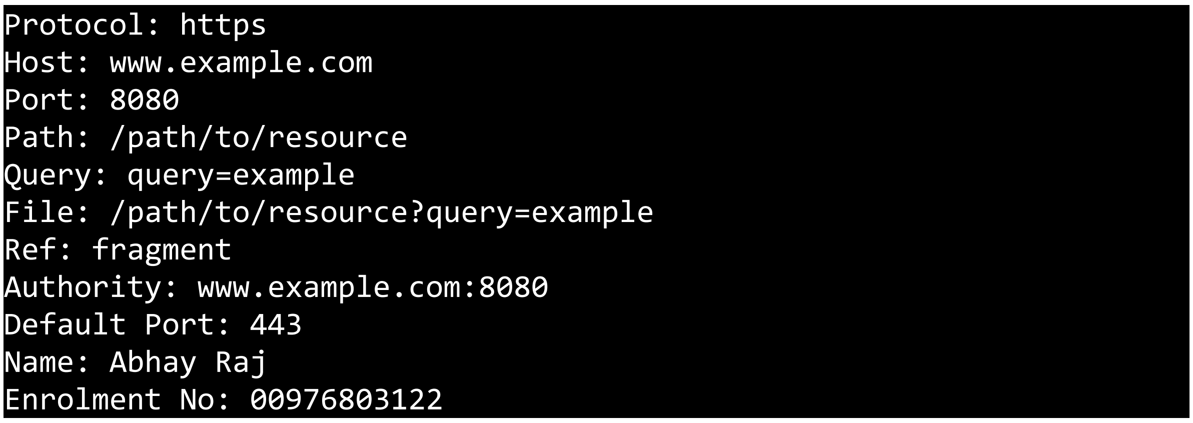
## Program – 11

Aim: Write a program to create a URL object and display its properties.

Code:

```java
import java.net.URL;

public class URLExample {
  public static void main(String[] args) {
    try {
      URL url = new URL("https://www.example.com:8080/path/to/resource?query=example#fragment");
      System.out.println("Protocol: " + url.getProtocol());
      System.out.println("Host: " + url.getHost());
      System.out.println("Port: " + url.getPort());
      System.out.println("Path: " + url.getPath());
      System.out.println("Query: " + url.getQuery());
      System.out.println("File: " + url.getFile());
      System.out.println("Ref: " + url.getRef());
      System.out.println("Authority: " + url.getAuthority());
      System.out.println("Default Port: " + url.getDefaultPort());
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Output:



```
Protocol: https
Host: www.example.com
Port: 8080
Path: /path/to/resource
Query: query=example
File: /path/to/resource?query=example
Ref: fragment
Authority: www.example.com:8080
Default Port: 443
Name: Abhay Raj
Enrolment No: 00976803122
```

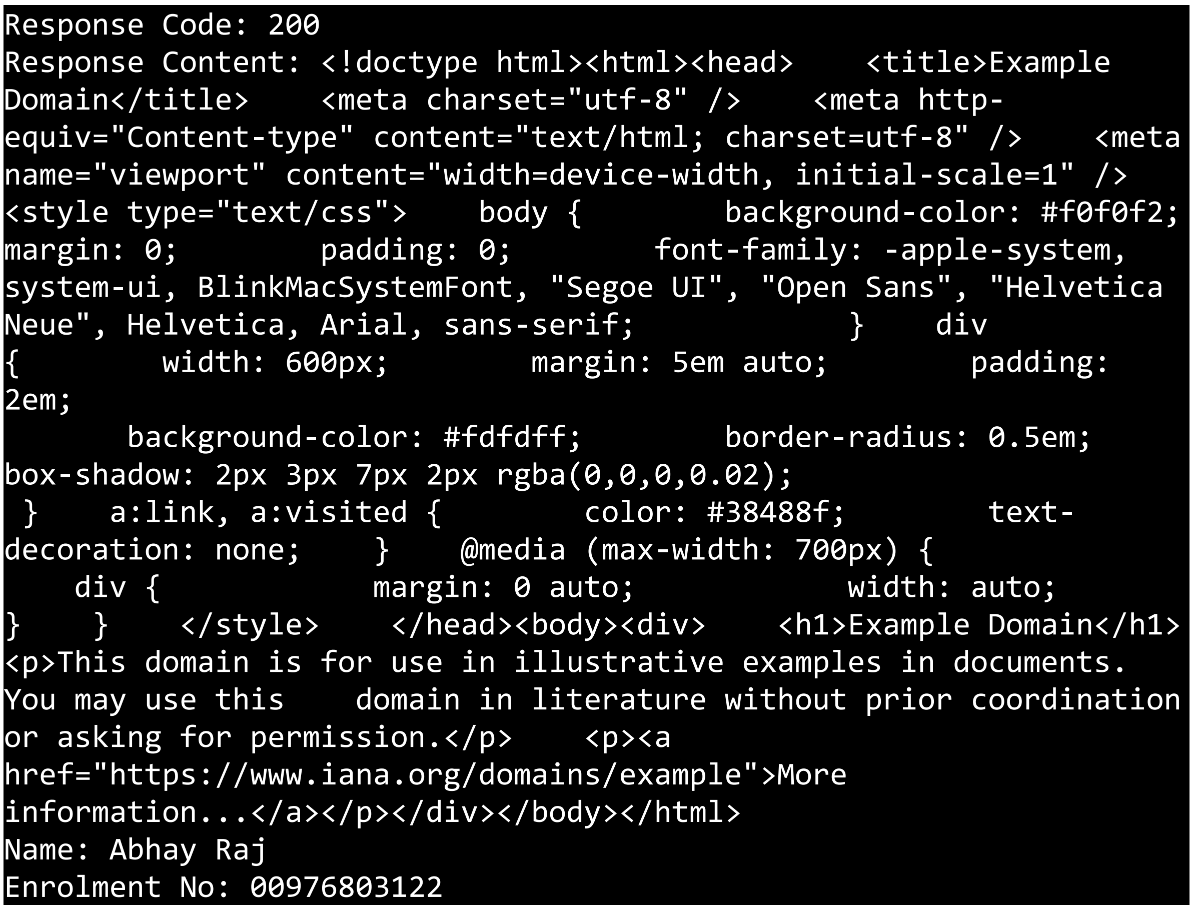
## Program – 12

Aim: Write a program to implement opening of a URL Connection.

Code:

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;

public class URLConnectionExample {
  public static void main(String[] args) {
    try {
      URL url = new URL("https://www.example.com");
      HttpURLConnection connection = (HttpURLConnection)
url.openConnection();

      connection.setRequestMethod("GET");
      connection.setConnectTimeout(5000);  // Set timeout to 5 seconds
      connection.setReadTimeout(5000);     // Set read timeout to 5 seconds

      int responseCode = connection.getResponseCode();
      System.out.println("Response Code: " + responseCode);

      if (responseCode == HttpURLConnection.HTTP_OK) {
        BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }
        in.close();

        System.out.println("Response Content: " + response.toString());
      } else {
        System.out.println("Failed to open connection.");
      }

      connection.disconnect();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Output:



```
Response Code: 200
Response Content: <!doctype html><html><head>    <title>Example
Domain</title>    <meta charset="utf-8" />    <meta http-
equiv="Content-type" content="text/html; charset=utf-8" />    <meta
name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">    body {        background-color: #f0f0f2;
margin: 0;        padding: 0;        font-family: -apple-system,
system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica
Neue", Helvetica, Arial, sans-serif;            }    div
{        width: 600px;        margin: 5em auto;        padding:
2em;
       background-color: #fdfdff;        border-radius: 0.5em;
box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
 }    a:link, a:visited {        color: #38488f;        text-
decoration: none;    }    @media (max-width: 700px) {
    div {            margin: 0 auto;            width: auto;
}    }    </style>    </head><body><div>    <h1>Example Domain</h1>
<p>This domain is for use in illustrative examples in documents.
You may use this    domain in literature without prior coordination
or asking for permission.</p>    <p><a
href="https://www.iana.org/domains/example">More
information...</a></p></div></body></html>
Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 13

Aim: Write a program to implement reading data from URL

Code:

```
import java.io.*;
import java.net.URL;
import java.net.HttpURLConnection;

public class ReadDataFromURL {
  public static void main(String[] args) {
    try {
      URL url = new URL("https://www.example.com");
      HttpURLConnection connection = (HttpURLConnection)
url.openConnection();

      connection.setRequestMethod("GET");
      connection.setConnectTimeout(5000);
      connection.setReadTimeout(5000);

      int responseCode = connection.getResponseCode();
      System.out.println("Response Code: " + responseCode);

      if (responseCode == HttpURLConnection.HTTP_OK) {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuilder content = new StringBuilder();

        while ((inputLine = reader.readLine()) != null) {
          content.append(inputLine).append("\n");
        }

        reader.close();
        System.out.println("Content from URL:");
        System.out.println(content.toString());
      } else {
        System.out.println("Failed to retrieve data. Response Code: " +
responseCode);
      }

      connection.disconnect();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Output:

```
Response Code: 200
Content from URL:
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html;
charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont,
"Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-
serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in
documents. You may use this
```

```
    domain in literature without prior coordination or asking for
permission.</p>
    <p><a href="https://www.iana.org/domains/example">More
information...</a></p>
</div>
</body>
</html>

Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 14

Aim: Write a program using HttpURLConnection for sending HTTP requests.

Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpURLConnectionExample {

  public static void main(String[] args) {
    try {
      String urlString = "https://jsonplaceholder.typicode.com/posts";
      URL url = new URL(urlString);

      HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
      connection.setRequestMethod("POST");
      connection.setRequestProperty("Content-Type", "application/json");
      connection.setDoOutput(true);

      String jsonInputString = "{\"title\": \"foo\", \"body\": \"bar\",
\"userId\": 1}";

      try (OutputStream os = connection.getOutputStream()) {
        byte[] input = jsonInputString.getBytes("utf-8");
        os.write(input, 0, input.length);
      }

      int responseCode = connection.getResponseCode();
      System.out.println("Response Code: " + responseCode);

      try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
        String inputLine;
        StringBuilder response = new StringBuilder();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }

        System.out.println("Response: " + response.toString());
      }

      connection.disconnect();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Output:

```
Response Code: 201
Response: {  "title": "foo",  "body": "bar",  "userId": 1,  "id":
101}
Name: Abhay Raj
Enrolment No: 00976803122
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in
documents. You may use this
    domain in literature without prior coordination or asking for
permission.</p>
    <p><a href="https://www.iana.org/domains/example">More
information...</a></p>
</div>
</body>
</html>

Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 15

Aim: Write a program to Make an HTTP Post Request.

Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpPostRequest {

  public static void main(String[] args) {
    try {
      String urlString = "https://jsonplaceholder.typicode.com/posts";
      URL url = new URL(urlString);

      HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
      connection.setRequestMethod("POST");
      connection.setRequestProperty("Content-Type", "application/json");
      connection.setDoOutput(true);

      String jsonInputString = "{\"title\": \"foo\", \"body\": \"bar\",
\"userId\": 1}";

      try (OutputStream os = connection.getOutputStream()) {
        byte[] input = jsonInputString.getBytes("utf-8");
        os.write(input, 0, input.length);
      }

      int responseCode = connection.getResponseCode();
      System.out.println("Response Code: " + responseCode);

      try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
        String inputLine;
        StringBuilder response = new StringBuilder();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }

        System.out.println("Response: " + response.toString());
      }

      connection.disconnect();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Response Code: 201
Response: {  "title": "foo",  "body": "bar",  "userId": 1,  "id":
101}
Name: Abhay Raj
Enrolment No: 00976803122
```

## Program – 16

Aim: Write a program to read a file using HTTP URL Connection.

Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpURLConnectionFileRead {

  public static void main(String[] args) {
    try {
      String urlString =
"https://raw.githubusercontent.com/selva86/datasets/refs/heads/master/
yoga.txt";
      URL url = new URL(urlString);

      HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
      connection.setRequestMethod("GET");
      connection.setRequestProperty("Accept", "text/plain");

      int responseCode = connection.getResponseCode();
      System.out.println("Response Code: " + responseCode);

      if (responseCode == HttpURLConnection.HTTP_OK) {
        try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
          String inputLine;
          StringBuilder content = new StringBuilder();

          while ((inputLine = in.readLine()) != null) {
            content.append(inputLine).append("\n");
          }

          System.out.println("File Content:\n" + content.toString());
        }
      } else {
        System.out.println("Failed to retrieve the file.");
      }

      connection.disconnect();
      System.out.println("Name: Abhay Raj\nEnrolment No: 00976803122");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Output:

```
Response Code: 200
File Content:
The ancient yogic discipline is far more than a fitness method for
physical health or a psychological tool to achieve
peace and happiness.
    Wellness of body and mind, often touted as the primary benefit
of modern yoga practice, is merely a by-product of
becoming a fully balanced and vibrantly alive being.
    The Sanskrit word 'yoga' comes from the word 'yuj' which means,
'to unite'.
    Hence, yoga is the union of the individual with the whole of
existence, also commonly referred to as 'self-realization',
'nirvana', 'mukti',  or 'enlightenment'.
    Yoga also refers to the inner technology that will lead one to
this experience â?? a technology formulated from rigorous inner
observation, by ancient yogis over thousands of years.
    With their extraordinary perception and mastery over every
aspect of the human mechanism, these great yogis delved into their
own systems, uncovering the nature of the cosmos â?? a macrocosm of
the human system.
    Initially, yoga was imparted by the Adiyogi (the first yogi),
Shiva, over 15,000 years ago.
    It was Adiyogi who introduced to humanity the idea that one can
evolve beyond oneâ??s present level of existence.
    He poured his knowing into the legendary Sapta Rishis, or seven
sages, who took the tremendous possibility offered by the yogic
science to various parts of the world, including Asia, ancient
Persia, northern Africa, and South America.
    It is this fundamental yet sophisticated science of elevating
human consciousness that is the source of the worldâ??s spiritual
traditions, predating religion by many thousands of years.

Name: Abhay Raj
Enrolment No: 00976803122
```