TokenRing Code:

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

import java.util.Arrays;

class TokenRing {

    private boolean hasToken = false;

    public synchronized void requestToken() {

        while (!hasToken) {

            try {

                wait();

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            }

        }

    }

    public synchronized void giveToken() {

        hasToken = true;

        notify();

    }

    public synchronized void passToken() {

        hasToken = false;

        notify();

    }

}

public class TokenRingExample {

    private static int totalProcesses;

    private static int[] processIDs;

    private static TokenRing tokenRing = new TokenRing();

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the total number of processes: ");

        totalProcesses = scanner.nextInt();

        scanner.nextLine(); // Consume newline left after nextInt()

        processIDs = new int[totalProcesses];

        System.out.println("Enter process IDs separated by space:");

        for (int i = 0; i < totalProcesses; i++) {

            processIDs[i] = scanner.nextInt();

        }

        scanner.nextLine(); // Consume newline left after loop

        Arrays.sort(processIDs);

        TokenPassing(scanner);

    }

    private static void TokenPassing(Scanner scanner) {

        tokenRing.giveToken();

        int i = 0;

        while (i < totalProcesses) {

            int currentProcessID = processIDs[i];

            String theRing = "";

            int tokenIndex;

            System.out.print("Do you want to pass the token to the next process? (y/n): ");

            String passOrComplete = scanner.nextLine();

            if (passOrComplete.equalsIgnoreCase("n")) {

                tokenIndex = (i + 2) % totalProcesses;

            } else {

                tokenIndex = (i + 1) % totalProcesses;

            }

            tokenRing.requestToken();

            System.out.println("Process " + currentProcessID + " is in the critical section.");

            try {

                Thread.sleep(1000);  // Simulate critical section

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            }

            if (passOrComplete.equalsIgnoreCase("n")) {

                System.out.println("Process " + currentProcessID + " completed and did not pass the token.");

            } else if (passOrComplete.equalsIgnoreCase("y")) {

                // Ask the user to enter a string to write to the shared file

                System.out.print("Enter a string to append to the log file for Process " + currentProcessID + ": ");

                String userInput = scanner.nextLine();  // Now reads full input including after Enter

                // Append the string to a single shared file

                try (FileWriter writer = new FileWriter("process\_log.txt", true)) {

                    writer.write("Process " + currentProcessID + ": " + userInput + "\n");

                    System.out.println("String appended to process\_log.txt");

                } catch (IOException e) {

                    System.out.println("An error occurred while writing to the file.");

                }

                // Pass the token to the next process

                int nextIndex = (i + 1) % totalProcesses;

                int nextProcessID = processIDs[nextIndex];

                System.out.println("Process " + currentProcessID + " passed the token to Process " + nextProcessID);

            } else {

                System.out.println("Invalid input, please enter 'y' or 'n'.");

                continue;

            }

            System.out.println(theRing);

            tokenRing.passToken();

            tokenRing.giveToken();

            try {

                Thread.sleep(1000);  // Simulate token passing delay

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

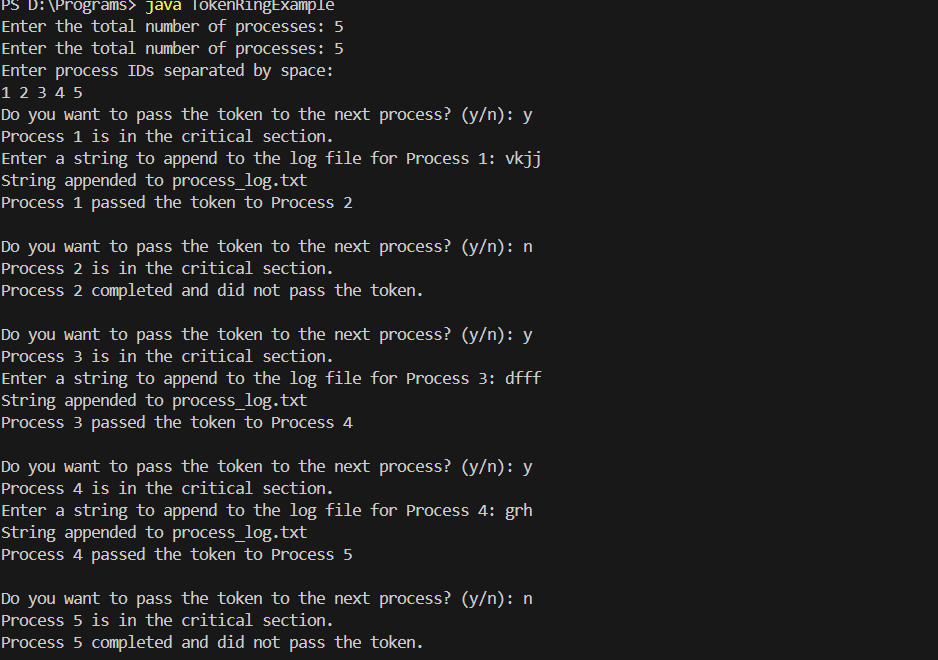
            }

            i++;

        }

    }

}



Process\_log.txt(Output):

Process 1: vkjj

Process 3: dfff

Process 4: grh