Below is a full, detailed explanation of the Ring.java program from your Assignment 6. This includes:

* 🔍 Line-by-line breakdown of the code
* 📘 Explanation of Ring Algorithm theory
* 🧠 Viva-style professor questions with answers
* 🧾 Each module’s role and logic
* ✅ Ready to be combined with your existing assignment PDF

**🌀 Ring Algorithm for Leader Election**

**📖 What is the Ring Algorithm?**

The Ring algorithm is used for electing a coordinator (leader) in a distributed system arranged in a logical ring. Here's how it works:

* All processes are arranged in a logical ring.
* Any process can start an election by sending a message to its neighbor.
* The message circulates, collecting IDs of active processes.
* The process with the highest ID becomes the coordinator.

**💻 Ring.java – Line-by-Line Explanation**

import java.util.Scanner;

Used to take user input (number of processes, process IDs, etc.).

public class Ring {

Defines the main class for implementing the Ring algorithm.

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Starts the main method and initializes Scanner.

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

int num = in.nextInt();

Asks user how many processes will participate.

Rr[] proc = new Rr[num];

Creates an array of process objects using a custom class Rr.

**🏗️ Initializing the Processes**

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

proc[i] = new Rr();

proc[i].index = i;

System.out.println("Enter the ID of process " + (i + 1) + ": ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

* Loops through each index and creates a process object.
* Takes ID input from user.
* Initializes state to active and flag (f) to 0 (used for tracking participation in election).

**🔁 Sorting Processes by ID**

for (int i = 0; i < num - 1; i++) {

for (int j = 0; j < num - 1; j++) {

if (proc[j].id > proc[j + 1].id) {

Rr temp = proc[j];

proc[j] = proc[j + 1];

proc[j + 1] = temp;

}

}

}

Sorts all processes by ID in ascending order using bubble sort, so that the highest ID is the last.

**📋 Display Sorted Process IDs**

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

System.out.print("[" + i + "] " + proc[i].id + " ");

}

Displays the index and ID of each process.

**👑 Assigning Initial Coordinator**

proc[num - 1].state = "inactive";

System.out.println("\nProcess " + proc[num - 1].id + " selected as coordinator");

The last process (with the highest ID) is initially set as the coordinator by marking it inactive (indicates leadership).

**📋 Main Menu for User**

while (true) {

System.out.println("\n1. Election\n2. Quit");

int ch = in.nextInt();

Displays options: start election or quit.

**🔄 Resetting Flags**

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

proc[i].f = 0;

}

Resets election participation flags for all processes.

**🗳️ Handling Election**

switch (ch) {

case 1:

System.out.println("Enter the process number that initializes the election: ");

int init = in.nextInt();

int temp2 = init;

int temp1 = init + 1;

int i = 0;

* User selects a process to initiate the election.
* temp1 and temp2 are used to move through the ring.

while (temp2 != temp1) {

if (temp1 == num) {

temp1 = 0;

}

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

i++;

}

temp1++;

}

* Continues message passing in a ring.
* Only active processes participate.
* Each participant sets its flag to 1.

**🏁 Coordinator Selection**

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

int max = -1;

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

if (max < proc[j].id) {

max = proc[j].id;

}

}

Finds the highest process ID (new coordinator).

System.out.println("Process " + max + " selected as coordinator");

for (int k = 0; k < num; k++) {

if (proc[k].id == max) {

proc[k].state = "inactive";

}

}

break;

Updates the new coordinator.

**❌ Exit Option**

case 2:

System.out.println("Program terminated.");

in.close();

return;

default:

System.out.println("Invalid response.");

break;

Handles user exit or invalid input.

**🧱 Rr Class – Process Structure**

class Rr {

public int index;

public int id;

public int f;

public String state;

}

* index: Position in array
* id: Process ID
* f: Flag for election participation
* state: "active" or "inactive"

**🧠 Professor Viva Questions (Ring Algorithm)**

| **Question** | **Answer** |
| --- | --- |
| What is the Ring algorithm used for? | Electing a coordinator among distributed processes. |
| Why is it called Ring algorithm? | Processes are logically connected in a circular ring. |
| What happens if the current coordinator crashes? | Any active process can initiate a new election. |
| How is the new coordinator chosen? | The process with the highest ID among active ones. |
| Is this algorithm centralized? | No, it is decentralized. |
| What is the message complexity? | O(n), where n is number of active processes. |
| What if two processes start elections at the same time? | They both initiate, but eventually converge to one coordinator. |

**✅ Summary: Key Modules**

| **Module** | **Role** |
| --- | --- |
| main() | Runs the program loop and handles user input |
| Process Initialization | Accepts ID and sets up each process |
| Sorting | Orders processes by ID for easy selection |
| Election Logic | Sends messages around the ring and collects responses |
| Coordinator Selection | Chooses highest ID among active nodes |

Sure! Below is the full content for the Ring Algorithm section that you can copy-paste into your Word file to append to your previous Token Ring assignment.

**📘 Assignment 06: Ring Algorithm for Leader Election**

**📖 What is the Ring Algorithm?**

The Ring Algorithm is a distributed leader election method used in systems where processes are arranged in a logical ring. When a process detects the need to elect a new coordinator (due to failure or unavailability of the current one), it initiates an election message that circulates through the ring. Each active process appends its ID to the message and passes it along. When the message returns to the initiator, it selects the process with the highest ID as the new coordinator.

**💻 Ring.java – Detailed Line-by-Line Explanation**

This program simulates the Ring algorithm using Java. Here’s how it works step by step:

**🔹 Importing Scanner**

import java.util.Scanner;

Used to read input from the user.

**🔹 Declaring Main Class**

public class Ring {

Defines the main class that contains the logic of the ring-based election.

**🔹 Starting the main() method and initializing input**

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int num = in.nextInt();

Takes the number of processes from the user and sets up the environment.

**🔹 Declaring the process array**

Rr[] proc = new Rr[num];

Creates an array of Rr objects to represent the processes.

**🔹 Initializing processes**

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

proc[i] = new Rr();

proc[i].index = i;

System.out.println("Enter the ID of process " + (i + 1) + ": ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

Sets each process ID, index, active state, and initializes the election flag (f = 0).

**🔹 Sorting processes based on ID**

for (int i = 0; i < num - 1; i++) {

for (int j = 0; j < num - 1; j++) {

if (proc[j].id > proc[j + 1].id) {

Rr temp = proc[j];

proc[j] = proc[j + 1];

proc[j + 1] = temp;

}

}

}

Sorts processes in ascending order using bubble sort, so the highest ID appears last.

**🔹 Displaying sorted processes**

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

System.out.print("[" + i + "] " + proc[i].id + " ");

}

Prints the process ID and index for user reference.

**🔹 Assigning initial coordinator**

proc[num - 1].state = "inactive";

System.out.println("\nProcess " + proc[num - 1].id + " selected as coordinator");

The highest ID process is initially marked as the coordinator (made inactive).

**🔹 Menu loop to start election or quit**

while (true) {

System.out.println("\n1. Election\n2. Quit");

int ch = in.nextInt();

Displays options and takes user input.

**🔹 Resetting election flags**

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

proc[i].f = 0;

}

Flags are reset before starting each election.

**🔹 Starting an election**

System.out.println("Enter the process number that initializes the election: ");

int init = in.nextInt();

int temp2 = init;

int temp1 = init + 1;

int i = 0;

while (temp2 != temp1) {

if (temp1 == num) {

temp1 = 0;

}

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

i++;

}

temp1++;

}

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

Simulates the circular election message passing among active processes.

**🔹 Electing the coordinator**

int max = -1;

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

if (max < proc[j].id) {

max = proc[j].id;

}

}

System.out.println("Process " + max + " selected as coordinator");

for (int k = 0; k < num; k++) {

if (proc[k].id == max) {

proc[k].state = "inactive";

}

}

Identifies the process with the highest ID and sets it as the new coordinator.

**🔹 Exiting the program**

case 2:

System.out.println("Program terminated.");

in.close();

return;

Handles user exit.

**🔧 Rr Class – Process Structure**

class Rr {

public int index; // Index in the process array

public int id; // Unique ID of the process

public int f; // Flag to track election participation

public String state; // "active" or "inactive"

}

**🎓 Viva Questions – Ring Algorithm**

Q: What is the Ring algorithm used for?  
A: Electing a coordinator among distributed processes.

Q: Why is it called Ring algorithm?  
A: Because processes are connected in a circular, logical ring.

Q: What happens if the current coordinator crashes?  
A: Any active process can initiate a new election.

Q: How is the new coordinator chosen?  
A: The active process with the highest ID.

Q: What is the message complexity?  
A: O(n), where n is the number of processes.

Q: Can two processes start an election simultaneously?  
A: Yes, but eventually one will win (highest ID) and all others will agree on the result.

Q: Is this a centralized algorithm?  
A: No, it's decentralized.