**Assignment No.**: 5  
**Assignment Name**: Creating Multiple Threads and Demonstrating Thread Communication

**Java Program: Multiple Threads with Communication**

In this example, we'll create two threads: one will produce a value, and the other will consume it. The threads will communicate using the wait() and notify() methods, which are part of Java's Object class.

// Class that represents a shared resource for communication between threads

class SharedResource {

private int data;

private boolean dataAvailable = false;

// Method to produce data

public synchronized void produceData(int value) throws InterruptedException {

// Wait until data is consumed

while (dataAvailable) {

wait(); // Wait for the consumer to consume the data

}

// Producing data

data = value;

System.out.println("Produced data: " + data);

dataAvailable = true; // Mark data as available for consumption

notify(); // Notify the consumer that data is available

}

// Method to consume data

public synchronized void consumeData() throws InterruptedException {

// Wait until data is available to consume

while (!dataAvailable) {

wait(); // Wait for the producer to produce data

}

// Consuming data

System.out.println("Consumed data: " + data);

dataAvailable = false; // Mark data as consumed

notify(); // Notify the producer that data has been consumed

}

}

// Producer thread class

class Producer extends Thread {

private SharedResource sharedResource;

public Producer(SharedResource sharedResource) {

this.sharedResource = sharedResource;

}

@Override

public void run() {

try {

// Produce multiple data values

for (int i = 1; i <= 5; i++) {

sharedResource.produceData(i);

Thread.sleep(1000); // Simulate time taken to produce data

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

// Consumer thread class

class Consumer extends Thread {

private SharedResource sharedResource;

public Consumer(SharedResource sharedResource) {

this.sharedResource = sharedResource;

}

@Override

public void run() {

try {

// Consume data values

for (int i = 1; i <= 5; i++) {

sharedResource.consumeData();

Thread.sleep(1500); // Simulate time taken to consume data

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

// Main class to run the threads

public class ThreadCommunicationExample {

public static void main(String[] args) {

// Create the shared resource object

SharedResource sharedResource = new SharedResource();

// Create the producer and consumer threads

Producer producer = new Producer(sharedResource);

Consumer consumer = new Consumer(sharedResource);

// Start the threads

producer.start();

consumer.start();

}

}

**Output Example:**

Produced data: 1

Consumed data: 1

Produced data: 2

Consumed data: 2

Produced data: 3

Consumed data: 3

Produced data: 4

Consumed data: 4

Produced data: 5

Consumed data: 5