LAB # 5

Inter-Thread Communication:

OBJECTIVE

Develop an inter-thread user communication program by using synchronization.

Lab Task:

1. Design a simple program of concurrency by implementing the scenario of two account holders in a joint bank account. (Hint: Total amount will be 50000, if 'user A' wants to withdraw 45,000 and 'user B' wants to withdraw 20,000) Apply mechanism of synchronization e.g. Block or Method for handling accessibility of multi-threads:

```
class BankAccount {
    private int balance = 50000; // Initial balance
    public synchronized void withdraw(String user, int amount) {
        System.out.println(user + " is trying to withdraw: " + amount);
        if (balance >= amount) {
            System.out.println(user + " is processing the withdrawal...");
            try {
                // Simulate time delay in withdrawal processing
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage()); }
            balance -= amount;
            System.out.println(user + " completed withdrawal of " + amount);
            System.out.println("Remaining balance: " + balance + "\n");
        } else {
            System.out.println("Insufficient balance for " + user + ". Current balance: " + balance + "\n"); }  }}
class WithdrawalThread extends Thread {
    private BankAccount account;
    private String user;
    private int amount;

    public WithdrawalThread(BankAccount account, String user, int amount) {
        this.account = account;
        this.user = user;
        this.amount = amount;
    }
    @Override
    public void run() {
        account.withdraw(user, amount);  }}
public class JavaApplication19 {
    public static void main(String[] args) {
        BankAccount jointAccount = new BankAccount();

        WithdrawalThread userA = new WithdrawalThread(jointAccount, "User A", 45000);
        WithdrawalThread userB = new WithdrawalThread(jointAccount, "User B", 20000);
        userA.start();
        userB.start();
    }
```

```
run:
User A is trying to withdraw: 45000
User A is processing the withdrawal...
User A completed withdrawal of 45000
Remaining balance: 5000

User B is trying to withdraw: 20000
Insufficient balance for User B. Current balance: 5000
```

2. Create an inter thread communication program of printer job by implementing two threads, one for calculating the remaining pages in printer tray and other one will print the pages that are pending on queue. (Hint: If total pages are 10 and user sends job for 15 pages than print thread will be on wait and will be notified once available pages are equal or greater than printing pages).

```
class Printer {
    private int availablePages = 10;

    public synchronized void printPages(int pagesToPrint) {
        System.out.println(Thread.currentThread().getName() + " requested to print " + pagesToPrint + " pages.");
        while (availablePages < pagesToPrint) {
            System.out.println("Not enough pages to print. Available: " + availablePages);
            System.out.println("Waiting for pages to be refilled...\n");
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        availablePages -= pagesToPrint;
        System.out.println("Printing " + pagesToPrint + " pages...");
        System.out.println("Pages remaining after print: " + availablePages + "\n");
    }

    public synchronized void addPages(int newPages) {
        System.out.println(Thread.currentThread().getName() + " is adding " + newPages + " pages to the tray.");
        availablePages += newPages;
        System.out.println("Total pages available now: " + availablePages + "\n");
        notify();
    }}
class PrintJob extends Thread {
    private Printer printer;
    private int pagesToPrint;

    public PrintJob(Printer printer, int pagesToPrint) {
        this.printer = printer;
        this.pagesToPrint = pagesToPrint;
    }

    @Override
    public void run() {
        printer.printPages(pagesToPrint);
    }
```

```
    @Override
    public void run() {
        try {
            Thread.sleep(4000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        printer.addPages(10);
    }
}

public class JavaApplication19 {
    public static void main(String[] args) {
        Printer printer = new Printer();
        PrintJob printJob = new PrintJob(printer, 15);
        TrayManager trayManager = new TrayManager(printer);

        printJob.setName("PrintJobThread");
        trayManager.setName("TrayManagerThread");

        printJob.start();
        trayManager.start();
    }
}
```

OUTPUT:

```
PrintJobThread requested to print 15 pages.
Not enough pages to print. Available: 10
Waiting for pages to be refilled...

TrayManagerThread is adding 10 pages to the tray.
Total pages available now: 20

Printing 15 pages...
Pages remaining after print: 5
```