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:

CODE

```
class JointAccount {
    private double balance = 50000; // total initial balance
    // synchronized method to prevent race conditions
    synchronized void withdraw(String user, double amount) {
        System.out.println(user + " is attempting to withdraw $" + amount);
        if (balance >= amount) {
            System.out.println(user + " is proceeding with the withdrawal...");
            try {
                // simulate delay in processing
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
            balance -= amount;
            System.out.println(user + " completed the withdrawal of $" + amount);
        } else {
            System.out.println("Insufficient balance for " + user + ". Withdrawal denied!");
        }
        System.out.println("Remaining balance: $" + balance + "\n");
    }
}
// Thread class representing a user
class AccountUser extends Thread {
    JointAccount account;
    String userName;
    double amount;
    AccountUser(JointAccount account, String userName, double amount) {
        this.account = account;
        this.userName = userName;
        this.amount = amount;
    }
    @Override
    public void run() {
        account.withdraw(userName, amount);
    }
}
class JointAccountDemo {
    public static void main(String[] args) {
        JointAccount account = new JointAccount();
        // Two users trying to withdraw simultaneously
        AccountUser userA = new AccountUser(account, "User A", 45000);
        AccountUser userB = new AccountUser(account, "User B", 20000);
        // start both threads
        userA.start();
        userB.start();
    }
}
```

OUTPUT

```
User A is attempting to withdraw $45000.0
User A is proceeding with the withdrawal...
User A completed the withdrawal of $45000.0
Remaining balance: $5000.0

User B is attempting to withdraw $20000.0
Insufficient balance for User B. Withdrawal denied!
Remaining balance: $5000.0
```

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).

CODE

```java
class Printer {
    private int availablePages = 10; // initial pages in the printer tray
    // synchronized method for printing
    synchronized void printPages(int pagesToPrint) {
        System.out.println("Print request received for " + pagesToPrint + " pages.");
        // if not enough pages, wait for refill
        while (availablePages < pagesToPrint) {
            System.out.println("Not enough pages to print. Waiting for refill...");
            try {
                wait(); // wait until notified
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        // perform printing
        System.out.println("Printing " + pagesToPrint + " pages...");
        availablePages -= pagesToPrint;
        System.out.println("Printing complete. Remaining pages: " + availablePages);
    }
    // synchronized method for adding pages to tray
    synchronized void addPages(int newPages) {
        System.out.println("\nAdding " + newPages + " pages to printer tray...");
        availablePages += newPages;
        System.out.println("Total available pages now: " + availablePages);
        notify(); // notify waiting print thread
    }
}
// Thread that handles printing job
class PrintThread extends Thread {
    private Printer printer;
    private int pagesToPrint;
    PrintThread(Printer printer, int pagesToPrint) {
        this.printer = printer;
        this.pagesToPrint = pagesToPrint;
    }
    @Override
    public void run() {
        printer.printPages(pagesToPrint);
    }
```

```
    }
}
// Thread that handles adding pages to printer tray
class TrayManagerThread extends Thread {
    private Printer printer;
    private int pagesToAdd;
    TrayManagerThread(Printer printer, int pagesToAdd) {
        this.printer = printer;
        this.pagesToAdd = pagesToAdd;
    }
    @Override
    public void run() {
        try {
            // simulate delay before adding pages
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
        printer.addPages(pagesToAdd);
    }
}
class PrinterJobDemo {
    public static void main(String[] args) {
        Printer printer = new Printer();
        // Create two threads:
        PrintThread printJob = new PrintThread(printer, 15); // needs 15 pages
        TrayManagerThread trayManager = new TrayManagerThread(printer, 10); // adds 10 more after some
            time
        // Start both threads
        printJob.start();
        trayManager.start();
    }
}
```

OUTPUT

```
Print request received for 15 pages.
Not enough pages to print. Waiting for refill...

Adding 10 pages to printer tray...
Total available pages now: 20
Printing 15 pages...
Printing complete. Remaining pages: 5
```