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:

Task1.java | Task2.java

```
1  package Lab5;
2  public class Task1 {
3      class BankAccount {
4          private int balance = 50000;
5          public synchronized void withdraw(String user, int amount) {
6              System.out.println(user + " is attempting to withdraw: " + amount);
7              if (balance >= amount) {
8                  try {
9                      Thread.sleep(1000);
10                 } catch (InterruptedException e) {
11                     System.out.println(e);}
12                 balance -= amount;
13                 System.out.println(user + " successfully withdrew " + amount + ". Remaining balance: " + b
14             } else {
15                 System.out.println(user + " attempted to withdraw " + amount + ", but insufficient funds.
16     class User extends Thread {
17         private BankAccount account;
18         private String userName;
19         private int withdrawalAmount;
20         public User(BankAccount account, String userName, int withdrawalAmount) {
21             this.account = account;
22             this.userName = userName;
23             this.withdrawalAmount = withdrawalAmount;}
24         @Override
25         public void run() {
26             account.withdraw(userName, withdrawalAmount);}}
27     public static void main(String[] args) {
28         Task1 task1 = new Task1();
29         BankAccount bankAccount = task1.new BankAccount();
30         User user1 = task1.new User(bankAccount, "Khurram Raza", 45000);
31         User user2 = task1.new User(bankAccount, "2023F-BSE-006", 20000);
32         user1.start();
33         user2.start();}}
```

Output:

Problems | Javadoc | Declaration | Console | Coverage

```
<terminated> Task1 (6) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (Nov 17, 2025, 1:00:59 AM – 1:01:01 AM elapsed: 0:00:01.500) [pid: 1
Khurram Raza is attempting to withdraw: 45000
Khurram Raza successfully withdrew 45000. Remaining balance: 5000
2023F-BSE-006 is attempting to withdraw: 20000
2023F-BSE-006 attempted to withdraw 20000, but insufficient funds. Remaining 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).

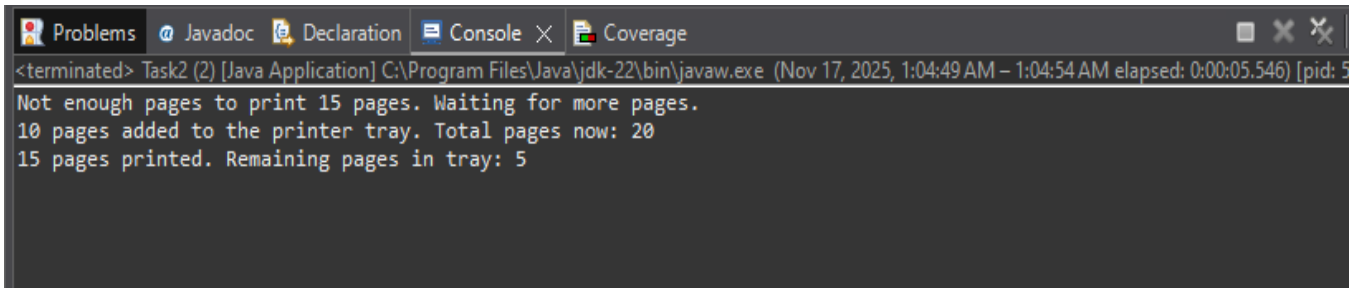
Code:

Task1.java Task2.java

```
package Lab5;
class Printer {
    private int pagesInTray = 10;
    public synchronized void addPages(int pages) {
        pagesInTray += pages;
        System.out.println(pages + " pages added to the printer tray. Total pages now: " + pagesInTray);
        notify();}
    public synchronized void printPages(int pages) throws InterruptedException {
        while (pagesInTray < pages) {
            System.out.println("Not enough pages to print " + pages + " pages. Waiting for more pages.");
            wait();}
        pagesInTray -= pages;
        System.out.println(pages + " pages printed. Remaining pages in tray: " + pagesInTray);}}
class PrintJobThread extends Thread {
    private Printer printer;
    private int pagesToPrint;
    public PrintJobThread(Printer printer, int pagesToPrint) {
        this.printer = printer;
        this.pagesToPrint = pagesToPrint;}
    @Override
    public void run() {
        try {
            printer.printPages(pagesToPrint);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());}}}
class RemainingPagesThread extends Thread {
    private Printer printer;
    public RemainingPagesThread(Printer printer) {
        this.printer = printer;}
    @Override
    public void run() {
        try {
            Thread.sleep(5000);
            printer.addPages(10);
        } catch (InterruptedException e) {
```

```
    @Override
    public void run() {
        try {
            Thread.sleep(5000);
            printer.addPages(10);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());}}}
public class Task2 {
    public static void main(String[] args) {
        Printer printer = new Printer();
        PrintJobThread printJob = new PrintJobThread(printer, 15);
        RemainingPagesThread remainingPages = new RemainingPagesThread(printer);
        printJob.start();
        remainingPages.start();}}
```

Output:



GitHub Screenshot: -