LAB # 5

Inter-Thread Communication

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:-

JoinAccount.java | Runable.java | Main.java

```
class JointAccount {  4 usages
    private int balance = 50000;  3 usages

    public synchronized void withdraw(String user, int amount) {  1 usage
        System.out.println(user + " is trying to withdraw: " + amount);

        if (balance >= amount) {
            System.out.println(user + " is processing withdrawal...");

            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            balance -= amount;
            System.out.println(user + " completed withdrawal of " + amount);
        } else {
            System.out.println("Insufficient balance for " + user + " to withdraw " + amount);
        }

        System.out.println("Remaining balance: " + balance);
        System.out.println("-----------------------------------");
    }
}
```

JoinAccount.java | Runable.java | Main.java

```
class WithdrawalTask implements Runnable {  2 usages
    private JointAccount account;  2 usages
    private String user;  2 usages
    private int amount;  2 usages

    public WithdrawalTask(JointAccount account, String user, int amount) {  2 usages
        this.account = account;
        this.user = user;
        this.amount = amount;
    }

    public void run() {
        account.withdraw(user, amount);
    }
}
```

JoinAccount.java Runable.java Main.java

```
public class Main {
    public static void main(String[] args) {
        JointAccount jointAccount = new JointAccount();

        Thread userA = new Thread(new WithdrawalTask(jointAccount, user: "User A", amount: 45000));
        Thread userB = new Thread(new WithdrawalTask(jointAccount, user: "User B", amount: 20000));

        userA.start();
        userB.start();
    }
}
```

Output:-

```
C:\Users\lab2\.jdks\openjdk-25\bin\java.exe "-jav
User A is trying to withdraw: 45000
User A is processing withdrawal...
User A completed withdrawal of 45000
Remaining balance: 5000
--------------------------------------
User B is trying to withdraw: 20000
Insufficient balance for User B to withdraw 20000
Remaining balance: 5000
--------------------------------------

Process finished with exit code 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:-

Printer.java PrintJobThread.java PageSupplierThread.java Main1.java

```
class Printer {  6 usages
    private int totalPages = 10;    5 usages

    public synchronized void printPages(int pagesToPrint) {  1 usage
        System.out.println("Print job requested for " + pagesToPrint + " pages.");

        while (totalPages < pagesToPrint) {
            System.out.println("Not enough pages to print. Waiting for refill...");
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        totalPages -= pagesToPrint;
        System.out.println("Printed " + pagesToPrint + " pages successfully.");
        System.out.println("Remaining pages in tray: " + totalPages);
        System.out.println("----------------------------------------");
    }

    public synchronized void addPages(int pages) {  1 usage
        System.out.println("Refilling printer tray with " + pages + " pages...");
        totalPages += pages;
        System.out.println("Pages available now: " + totalPages);

        notify();
        System.out.println("Notified waiting print job.");
        System.out.println("----------------------------------------");
    }
}
```

Printer.java | PrintJobThread.java | PageSupplierThread.java

```
class PrintJobThread extends Thread {  2 usages
    private Printer printer;  2 usages
    private int pages;  2 usages

    public PrintJobThread(Printer printer, int pages) {  1 usage
        this.printer = printer;
        this.pages = pages;
    }

    public void run() {
        printer.printPages(pages);
    }
}
```

Printer.java | PrintJobThread.java | PageSupplierThread.java | Main1.java

```
class PageSupplierThread extends Thread {  2 usages
    private Printer printer;  2 usages
    private int pagesToAdd;  2 usages

    public PageSupplierThread(Printer printer, int pagesToAdd) {  1 usage
        this.printer = printer;
        this.pagesToAdd = pagesToAdd;
    }

    public void run() {
        try {
            Thread.sleep( millis: 3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        printer.addPages(pagesToAdd);
    }
}
```

Printer.java | PrintJobThread.java | PageSupplierThread.java | Main1.java

```
public class Main1 {
    public static void main(String[] args) {
        Printer printer = new Printer();

        PrintJobThread printJob = new PrintJobThread(printer,  pages: 15);

        PageSupplierThread supplier = new PageSupplierThread(printer,  pagesToAdd: 10);

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

Output:-

```
C:\Users\lab2\.jdks\openjdk-25\bin\java.exe "-ja
Print job requested for 15 pages.
Not enough pages to print. Waiting for refill...
Refilling printer tray with 10 pages...
Pages available now: 20
Notified waiting print job.
-----------------------------------
Printed 15 pages successfully.
Remaining pages in tray: 5
-----------------------------------

Process finished with exit code 0
```