LAB#06
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

```
1   package Task1;
2   class BankAccount {
3       private int balance;
4
5       public BankAccount(int initialBalance) {
6           this.balance = initialBalance;
7       }
8
9       public synchronized void withdraw(int amount, String user) {
10          if (balance >= amount) {
11              System.out.println(user + " is withdrawing " + amount);
12              try {
13                  Thread.sleep(1000);
14              } catch (InterruptedException e) {
15                  System.out.println("Thread interrupted.");
16              }
17              balance -= amount;
18              System.out.println(user + " has withdrawn " + amount + ". New balance: " + balance);
19          } else {
20              System.out.println(user + " tried to withdraw " + amount + ",
21              but insufficient funds. Current balance: " + balance);
22          }
23      }
24
25      public int getBalance() {
26          return balance;
27      }
28  }
```

```java
package Task1;
public class JointAccountExample {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(50000);

        Thread userA = new Thread(() -> account.withdraw(45000, "User A"));
        Thread userB = new Thread(() -> account.withdraw(20000, "User B"));

        userA.start();
        userB.start();

        try {
            userA.join();
            userB.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        System.out.println("Final balance in the account: " + account.getBalance());
    }
}
```

Output:

```
User A is withdrawing 45000
User A has withdrawn 45000. New balance: 5000
User B tried to withdraw 20000, but insufficient funds. Current balance: 5000
Final balance in the account: 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:

```java
package Task2;
class Printer {
    private int pagesInTray;

    public Printer(int initialPages) {
        this.pagesInTray = initialPages;
    }
```

```
9         public synchronized void addPages(int pages) {
10            System.out.println("Adding " + pages + " pages to the tray.");
11            pagesInTray += pages;
12            System.out.println("Total pages in tray: " + pagesInTray);
13            notify();
14        }
15
16        public synchronized void printPages(int pages) {
17            System.out.println("Print job requested for " + pages + " pages.");
18            while (pages > pagesInTray) {
19                System.out.println("Not enough pages in the tray. Waiting for more pages...");
20                try {
21                    wait();
22                } catch (InterruptedException e) {
23                    System.out.println("Print thread interrupted.");
24                }
25            }
26            pagesInTray -= pages;
27            System.out.println("Printed " + pages + " pages. Remaining pages in tray: " + pagesInTray);
28        }
29    }
```

```
1     package Task2;
2     public class PrinterJob {
3         public static void main(String[] args) {
4             Printer printer = new Printer(10);
5
6             Thread printThread = new Thread(() -> {
7                 printer.printPages(15);
8             });
```

```
10            Thread addPagesThread = new Thread(() -> {
11                try {
12                    Thread.sleep(2000);
13                    printer.addPages(10);
14                } catch (InterruptedException e) {
15                    System.out.println("Add pages thread interrupted.");
16                }
17            });
18
19            printThread.start();
20            addPagesThread.start();
21
22            try {
23                printThread.join();
24                addPagesThread.join();
25            } catch (InterruptedException e) {
26                System.out.println("Main thread interrupted.");
27            }
```

```
29              System.out.println("All printer jobs completed.");
30          }
31      }
```

OUTPUT:

```
Not enough pages in the tray. Waiting for more pages...
Adding 10 pages to the tray.
Total pages in tray: 20
Printed 15 pages. Remaining pages in tray: 5
All printer jobs completed.
```

Github: