```java
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Semaphore;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;


public class ThreadCooperation {
    private static Account account = new Account();

    public static void main(String[] args) {
        ExecutorService executor =
Executors.newFixedThreadPool(2);
        executor.execute(new DepositTask());
        executor.execute(new WithdrawTask());
        executor.shutdown();
        System.out.println("Thread 1\t\tThread 2\t\tBalance");
    }

    public static class DepositTask implements Runnable {
        @Override
        public void run() {
            try {
                while(true) {
                    account.deposit((int)(Math.random()*10) +
1);
                    Thread.sleep(2000);
                }
            }
            catch(InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }

    public static class WithdrawTask implements Runnable {
        @Override
        public void run() {
            while(true) {
                account.withdraw((int)(Math.random() * 10) +
1);
            }
        }
```

```
    }

    private static class Account {
        private static Semaphore semaphore = new Semaphore(1);
        private int balance = 0;
        public int getBalance() {
            return balance;
        }

        public void withdraw(int amount) {
            while (balance <= amount) Thread.currentThread();
            try {
                semaphore.acquire();
                balance -= amount;
                System.out.println("\t\t\tWithdraw: " + amount
+
                        "\t\t" + getBalance());
            }
            catch (InterruptedException e) {
// TODO Auto-generated catch block
                e.printStackTrace();
            }
            finally {
                semaphore.release();
            }
        }

        public void deposit(int amount) {
            try {
                semaphore.acquire();
                balance += amount;
                System.out.println("Deposit: " + amount +
                        "\t\t\t\t\t" + getBalance());
            }
            catch (InterruptedException ex){
                ex.printStackTrace();
            }
            finally {
                semaphore.release();
            }
        }
    }
}
```

- Execution result

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applic
Thread 1        Thread 2        Balance
Deposit 5                       5
            Withdraw3           2
            Wait for a deposit
Deposit 1                       3
            Wait for a deposit
Deposit 7                       10
            Withdraw9           1
            Wait for a deposit
Deposit 7                       8
            Withdraw8           0
            Wait for a deposit
Deposit 4                       4
            Wait for a deposit
Deposit 6                       10
            Withdraw9           1
            Wait for a deposit
Deposit 2                       3
            Wait for a deposit
Deposit 6                       9
            Withdraw6           3
            Wait for a deposit
Deposit 10                      13
            Withdraw6           7
            Withdraw3           4
            Wait for a deposit
Deposit 3                       7
            Withdraw7           0
            Wait for a deposit
```