lockinresource.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# lockinresource.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()

    def get_balance(self):
        self._lock.acquire()
        try:
            return self._balance
        finally:
            self._lock.release()

    def deposit(self, amount):
        self._lock.acquire()
        temp = self._balance
        temp += amount
        print(temp)
        self._balance = temp
        self._lock.release()

    def withdraw(self, amount):
        self._lock.acquire()
        temp = self._balance
        temp -= amount
        print(temp)
        self._balance = temp
        self._lock.release()

#-----------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#-----------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#-----------------------------------------------------------------------
```

lockinresource.py (Page 2 of 2)

```

def main():

    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

lockinresourcew.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# lockinresourcew.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()

    def get_balance(self):
        with self._lock:
            return self._balance

    def deposit(self, amount):
        with self._lock:
            temp = self._balance
            temp += amount
            print(temp)
            self._balance = temp

    def withdraw(self, amount):
        with self._lock:
            temp = self._balance
            temp -= amount
            print(temp)
            self._balance = temp

#-----------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#-----------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#-----------------------------------------------------------------------

def main():

    bank_acct = BankAcct()

```

lockinresourcew.py (Page 2 of 2)

```
    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

conditions.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# conditions.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()
        self._condition = threading.Condition(self._lock)

    def get_balance(self):
        self._lock.acquire()
        try:
            return self._balance
        finally:
            self._lock.release()

    def deposit(self, amount):
        self._condition.acquire()
        self._balance += amount
        print(self._balance)
        self._condition.notify_all()
        self._condition.release()

    def withdraw(self, amount):
        self._condition.acquire()
        while self._balance < amount:
            self._condition.wait()
        self._balance -= amount
        print(self._balance)
        self._condition.release()

#-----------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#-----------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#-----------------------------------------------------------------------
```

conditions.py (Page 2 of 2)

```

def main():
    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

conditionsw.py (Page 1 of 2)

```
#!/usr/bin/env python

#------------------------------------------------------------------------
# conditionsw.py
# Author: Bob Dondero
#------------------------------------------------------------------------

import threading

#------------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()
        self._condition = threading.Condition(self._lock)

    def get_balance(self):
        with self._lock:
            return self._balance

    def deposit(self, amount):
        with self._condition:
            self._balance += amount
            print(self._balance)
            self._condition.notify_all()

    def withdraw(self, amount):
        with self._condition:
            while self._balance < amount:
                self._condition.wait()
            self._balance -= amount
            print(self._balance)

#------------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#------------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#------------------------------------------------------------------------

def main():
    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
```

conditionsw.py (Page 2 of 2)

```
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

prodconprocesses.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# prodconprocesses.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import multiprocessing

#-----------------------------------------------------------------------

def produce(producer_conn):
    for i in range(100):
        producer_conn.send(i)
        print('Produced:', i)

#-----------------------------------------------------------------------

def consume(consumer_conn):
    for _ in range(100):
        num = consumer_conn.recv()
        print('Consumed:', num)

#-----------------------------------------------------------------------

def main():

    producer_conn, consumer_conn = multiprocessing.Pipe()

    producer_process = multiprocessing.Process(
        target=produce, args=[producer_conn])
    consumer_process = multiprocessing.Process(
        target=consume, args=[consumer_conn])

    producer_process.start()
    consumer_process.start()

    producer_process.join()
    consumer_process.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```


prodconthreads.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# prodconthreads.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading
import queue

QUEUE_SIZE = 8  # Arbitrary

#-----------------------------------------------------------------------

class ProducerThread (threading.Thread):

    def __init__(self, intqueue):
        threading.Thread.__init__(self)
        self._intqueue = intqueue

    def run(self):
        for i in range(100):
            self._intqueue.put(i)
            print('Produced:', i)

#-----------------------------------------------------------------------

class ConsumerThread (threading.Thread):

    def __init__(self, intqueue):
        threading.Thread.__init__(self)
        self._intqueue = intqueue

    def run(self):
        for _ in range(100):
            num = self._intqueue.get()
            print('Consumed:', num)

#-----------------------------------------------------------------------

def main():

    intqueue = queue.Queue(QUEUE_SIZE)

    producer_thread = ProducerThread(intqueue)
    consumer_thread = ConsumerThread(intqueue)

    producer_thread.start()
    consumer_thread.start()

    producer_thread.join()
    consumer_thread.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

Conditions.java (Page 1 of 2)

```
 1: //-------------------------------------------------------------------------
 2: // Conditions.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: class Bank
 7: {
 8:    private int balance = 0;
 9:
10:    public synchronized int getBalance()
11:    {
12:       return balance;
13:    }
14:
15:    public synchronized void deposit(int amount)
16:    {
17:       balance += amount;
18:       System.out.println(balance);
19:       notifyAll();
20:    }
21:
22:    public synchronized void withdraw(int amount)
23:    {
24:       try
25:       {
26:          while (balance < amount)
27:             wait();
28:          balance -= amount;
29:          System.out.println(balance);
30:       }
31:       catch (InterruptedException e)
32:       {
33:          Thread.currentThread().interrupt();
34:       }
35:    }
36: }
37:
38: //-------------------------------------------------------------------------
39:
40: class DepositThread extends Thread
41: {
42:    private Bank bank;
43:
44:    public DepositThread(Bank bank)
45:    {
46:       this.bank = bank;
47:    }
48:
49:    public void run()
50:    {
51:       for (int i = 0; i < 10; i++)
52:          bank.deposit(1);
53:    }
54: }
55:
56: //-------------------------------------------------------------------------
57:
58: class WithdrawThread extends Thread
59: {
60:    private Bank bank;
61:
62:    public WithdrawThread(Bank bank)
63:    {
64:       this.bank = bank;
65:    }
```

Conditions.java (Page 2 of 2)

```
66:
67:    public void run()
68:    {
69:       for (int i = 0; i < 5; i++)
70:          bank.withdraw(2);
71:    }
72: }
73:
74: //-------------------------------------------------------------------------
75:
76: public class Conditions
77: {
78:    public static void main(String[] args)
79:    {
80:       Bank bank = new Bank();
81:
82:       Thread depositThread = new DepositThread(bank);
83:       Thread withdrawThread = new WithdrawThread(bank);
84:
85:       depositThread.start();
86:       withdrawThread.start();
87:
88:       try
89:       {
90:          depositThread.join();
91:          depositThread.join();
92:       }
93:       catch (InterruptedException e)
94:       {
95:          Thread.currentThread().interrupt();
96:       }
97:
98:       System.out.println("Final balance: " + bank.getBalance());
99:    }
100: }
```

conditions.c (Page 1 of 2)

```
 1: /*-------------------------------------------------------------------------*/
 2: /* conditions.c                                                            */
 3: /* Author: Bob Dondero                                                     */
 4: /* To build use the -pthread option                                        */
 5: /*-------------------------------------------------------------------------*/
 6:
 7: #include <pthread.h>
 8: #include <stdio.h>
 9: #include <stdlib.h>
10:
11: /*-------------------------------------------------------------------------*/
12:
13: /* Think of this as a BankAcct object. */
14:
15: static int bankAcctBalance = 0;
16: static pthread_mutex_t bankAcctMutex;
17: static pthread_cond_t bankAcctCondition;
18:
19: static void deposit(int amount)
20: {
21:    pthread_mutex_lock(&bankAcctMutex);
22:    bankAcctBalance += amount;
23:    printf("%d\n", bankAcctBalance);
24:    pthread_cond_signal(&bankAcctCondition);
25:    pthread_mutex_unlock(&bankAcctMutex);
26: }
27:
28: static void withdraw(int amount)
29: {
30:    pthread_mutex_lock(&bankAcctMutex);
31:    while (bankAcctBalance < amount)
32:       pthread_cond_wait(&bankAcctCondition, &bankAcctMutex);
33:    bankAcctBalance -= amount;
34:    printf("%d\n", bankAcctBalance);
35:    pthread_mutex_unlock(&bankAcctMutex);
36: }
37:
38: /*-------------------------------------------------------------------------*/
39:
40: static void *makeDeposits()
41: {
42:    int i;
43:    for (i = 0; i < 10; i++)
44:       deposit(1);
45:    return NULL;
46: }
47:
48: /*-------------------------------------------------------------------------*/
49:
50: static void *makeWithdrawals()
51: {
52:    int i;
53:    for (i = 0; i < 5; i++)
54:       withdraw(2);
55:    return NULL;
56: }
57:
58: /*-------------------------------------------------------------------------*/
59:
60: int main(void)
61: {
62:    pthread_t depositThread;
63:    pthread_t withdrawThread;
64:
65:    pthread_mutex_init(&bankAcctMutex, NULL);
```

conditions.c (Page 2 of 2)

```
66:    pthread_cond_init(&bankAcctCondition, NULL);
67:
68:    pthread_create(&depositThread, NULL, makeDeposits, NULL);
69:    pthread_create(&withdrawThread, NULL, makeWithdrawals, NULL);
70:
71:    pthread_join(depositThread, NULL);
72:    pthread_join(withdrawThread, NULL);
73:
74:    pthread_mutex_destroy(&bankAcctMutex);
75:    pthread_cond_destroy(&bankAcctCondition);
76:
77:    printf("Final bankAcctBalance: %d\n", bankAcctBalance);
78:    return 0;
79: }
```