package AdderSubtractor.Mutex;  
  
public class Count {  
 int value = 0;  
}

package AdderSubtractor.Mutex;  
  
import java.util.concurrent.locks.Lock;  
  
public class Adder implements Runnable{  
 //Data member  
 Count count;  
 Lock lock;  
  
 //Constructor  
 public Adder(Count count, Lock lock){  
 this.count = count;  
 this.lock = lock;  
 }  
  
 @Override  
 public void run() {  
 for(int i =1; i<=1000000; i++){  
 synchronized (count){  
 count.value += i;  
 }  
 }  
 }  
}

package AdderSubtractor.Mutex;  
  
import java.util.concurrent.locks.Lock;  
  
public class Subtractor implements Runnable{  
 //data member  
 Count count;  
 Lock lock;  
  
 //constructor  
 public Subtractor(Count count, Lock lock){  
 this.count =count;  
 this.lock = lock;  
 }  
 @Override  
 public void run() {  
 for(int i =1; i<=1000000; i++){  
 synchronized (count){  
 count.value -= i;  
 }  
 }  
 }  
}

package AdderSubtractor.Mutex;  
  
import java.util.concurrent.locks.Lock;  
import java.util.concurrent.locks.ReentrantLock;  
  
public class Client {  
 public static void main(String[] args) throws InterruptedException {  
 Count count = new Count();  
 Lock lock = new ReentrantLock();  
  
 Adder adder = new Adder(count, lock);  
 Subtractor subtractor = new Subtractor(count, lock);  
  
 // Thread  
 Thread t1 = new Thread(adder);  
 Thread t2 = new Thread(subtractor);  
  
 t1.start();  
 t2.start();  
  
 //wait for t1 and t2 complete  
 t1.join();  
 t2.join();  
  
 System.*out*.println("Both the threads have completed Count : "+ count.value);  
 }  
}