|  |
| --- |
| 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

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |