```
/*****************************************************************************
/
/      filename:  Monitor.java
/
/   description: This program creates a monitor class that takes 3 different
/                thread types and exicutes them accordingly.
/
/        author:  Paladino, Zac
/      login id:  cps346-01.16
/
/         class:  CPS 346
/    instructor:  Perugini
/    assignment:  Homework #5
/
/      assigned:  March 10, 2009
/           due:  March 17, 2009
/
*****************************************************************************/

import java.util.concurrent.*;
import java.util.concurrent.locks.*;

// ME among A's.
// ME between B's and C's.
// ME between A's and C's.
// Priority to A's.

class Monitor {

   Lock lock = new ReentrantLock();
   Condition ConA = lock.newCondition();
   Condition ConB = lock.newCondition();
   Condition ConC = lock.newCondition();
   boolean ain, bin, cin;
   int as,bs,cs;

   Monitor() {
     ain = false;
     bin = false;
     cin = false;
     as = 0;
     bs = 0;
     cs = 0;
   }

   public void Aentry() {
      lock.lock();

      as++;
      if(ain || cin){
        try{ConA.await();}catch(InterruptedException e){}
      }

      ain = true;
      lock.unlock();
   }

   public void Aexit() {
      lock.lock();
      ain = false;
      as--;
      if(as > 0){
        ConA.signal();
      }
      else if(bs > 0){
        ConB.signal();
      }
      else if(cs > 0){
        ConC.signal();
      }
      lock.unlock();
   }

   public void Bentry() {
      lock.lock();

      bs++;
      if(as > 0 || cin){
        try{ConB.await();}catch(InterruptedException e){}
      }

      bin = true;
      lock.unlock();
   }

   public void Bexit() {
      lock.lock();
      bs--;
      bin = false;
      if(as > 0){
        ConA.signal();
      }
      else if(bs > 0){
        ConB.signal();
      }
      else if(cs > 0){
        ConC.signal();
      }
      lock.unlock();
   }

   public void Centry() {
      lock.lock();

      cs++;
      if(as > 0 || ain || bin){
        try{ConC.await();}catch(InterruptedException e){}
      }

      cin = true;
      lock.unlock();
   }

   public void Cexit() {
      lock.lock();
      cs--;
      cin = false;
      if(as > 0){
        ConA.signal();
      }
      else if(bs > 0){
        ConB.signal();
      }
      else if(cs > 0){
        ConC.signal();
      }
      lock.unlock();
   }
}
```