```
/* ENTREGABLE 3 - OPCIÓN 1 */

volatile static boolean en_sc = false;

static class Incrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    while(en_sc) { }
                    en_sc = true;

                    // Seccion critica
                    sc_inc();

                    // Protocolo de salida de la seccion critica
                    en_sc = false;
               }
          }
     }

     static class Decrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    while(en_sc) { }
                    en_sc = true;

                    // Seccion critica
                    sc_dec();

                    // Protocolo de salida de la seccion critica
                    en_sc = false;
               }
          }
     }
```

Esta solución NO garantiza la exclusión mutua, si ambos procesos ejecutan de manera simultánea el while(en_sc){}, entonces pasa a la siguiente sentencia y acaba produciendo una condición de carrera.

```
/* ENTREGABLE 3 - OPCIÓN 2 */

     volatile static boolean en_sc = false;
     volatile static boolean turno_inc = true;

     static class Incrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    while(!turno_inc || en_sc) { }
                    en_sc = true;

                    // Seccion critica
                    sc_inc();

                    // Protocolo de salida de la seccion critica
                    en_sc = false;
                    turno_inc = false;
               }
          }
     }

     static class Decrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    while(turno_inc || en_sc) { }
                    en_sc = true;

                    // Seccion critica
                    sc_dec();

                    // Protocolo de salida de la seccion critica
                    en_sc = false;
                    turno_inc = true;
               }
          }
     }
```

En este caso se podrían producir esperas innecesarias al intentar ejecutar el while del Decrementador, ya que hasta que no se

ejecute (al menos) una iteración del Incrementador, no podrá salir del bucle

```
/* ENTREGABLE 3 - OPCIÓN 3 */

     volatile static boolean en_sc_inc = false;
     volatile static boolean en_sc_dec = false;

     static class Incrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    en_sc_inc = true;
                    while(en_sc_dec) { }

                    // Seccion critica
                    sc_inc();

                    // Protocolo de salida de la seccion critica
                    en_sc_inc = false;
               }
          }
     }


     static class Decrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    en_sc_dec = true;
                    while(en_sc_inc) { }

                    // Seccion critica
                    sc_dec();

                    // Protocolo de salida de la seccion critica
                    en_sc_dec = false;
               }
          }
     }
```

En este caso si se ejecutan las sentencias en_sc_dec = true; y en_sc_inc = true; antes de avanzar a comprobar el while, entonces se producirá un interbloqueo, ya que el programa no será capaz de avanzar a la siguiente ejecución (salir del bucle while)

```
/* ENTREGABLE 3 - OPCIÓN 4 → CORRECTA */

     volatile static boolean en_sc_inc = false;
     volatile static boolean en_sc_dec = false;
     volatile static boolean turno_inc = false;

     static class Incrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    en_sc_inc = true;
                    turno_inc = false;
                    while(en_sc_dec && !turno_inc) { }

                    // Seccion critica
                    sc_inc();

                    // Protocolo de salida de la seccion critica
                    en_sc_inc = false;

               }
          }
     }

     static class Decrementador extends Thread {
          public void run() {
               for(int i = 0; i < N_PASOS; i++) {
                    // Seccion no critica
                    no_sc();

                    // Protocolo de acceso a la seccion critica
                    en_sc_dec = true;
                    turno_inc = true;
                    while(en_sc_inc && turno_inc) { }

                    // Seccion critica
                    sc_dec();

                    // Protocolo de salida de la seccion critica
                    en_sc_dec = false;

               }
          }
     }
```