Práctica 2

- Primera versión

Monitor

np : int = 0 ⇝ número de peatones

ncd : int = 0 ⇝ número de coches en sentido derecho

nci : int = 0 ⇝ número de coches en sentido izquierdo

Invariantes

$$\left\{ \begin{array}{l} np, ncd, nci \geq 0 \\ np > 0 \Rightarrow ncd = 0 \wedge nci = 0 \\ ncd > 0 \Rightarrow nci = 0 \end{array} \right\} \equiv INV$$

Métodos

```
enter_np()
    wait(ncd==0 ∧ nci==0)
    np = np+1

leave_np()
    np = np-1
```

```
enter_ncd()
    wait(np==0 ∧ nci==0)
    ncd = ncd+1

leave_ncd()
    ncd = ncd-1
```

```
enter_nci()
    wait(np==0 ∧ ncd==0)
    nci = nci+1

leave_nci()
    nci = nci-1
```

Variables condición

ncd==0 ∧ nci==0 ⇝ nocoches : vc = True

np==0 ∧ nci==0 ⇝ nop_noci : vc = True

np==0 ∧ ncd==0 ⇝ nop_nocd : vc = True.

Los métodos quedarían:

```
enter_np()
    {INV}
    nocoches.wait(ncd==0 ∧ nci==0)
    {INV ∧ ncd==0 ∧ nci==0}
    np = np+1
    {INV ∧ np>0}
```

```
enter_ncd()
    {INV}
    nop_noci.wait(np==0 ∧ nci==0)
    {INV ∧ np==0 ∧ nci==0}
    ncd = ncd+1
    {INV ∧ ncd>0}
```

```
enter_nci()
  {INV}
  nop_ncd.wait(np==0 ∧ ncd==0
  {INV ∧ np==0 ∧ ncd==0}
  nci = nci + 1
  {INV ∧ nci > 0}
```

```
leave_np()
{INV ∧ np > 0}
  np = np - 1
  if np == 0:
     nop_ncd.notify()
     nop_nci.notify()
{INV}
```

```
leave_ncd()
  {INV ∧ ncd > 0}
  ncd = ncd - 1
  if ncd == 0:
     nocoches.notify()
     nop_ncd.notify()
  {INV}
```

```
leave_nci()
  {INV ∧ nci > 0}
  nci = nci - 1
  if nci == 0:
     nocoches.notify()
     nop_nci.notify()
  {INV}
```

- Problema de inanición

Hay varias situaciones que pueden llevar a un problema de inanición:

(i) Que haya muchos peatones y empiecen a pasar mientras que los coches se queden esperando

(ii) Que los peatones y los coches en sentido a la derecha se turnen para pasar, y los coches en sentido a la izquierda se queden esperando

(iii) Que los peatones y los coches en sentido a la izquierda se turnen para pasar, y los coches en sentido a la derecha se queden esperando

(iv) Que los coches en sentido a la derecha y a la izquierda se turnen para pasar, y los peatones se queden esperando.

Solución: Vamos a añadir tres variables que cuenten cuántos de cada clase están esperando:

```
np_waiting : int = 0 ; ncd_waiting : int = 0 ; nci_waiting : int = 0
p_waiting : cv       ; cd_waiting : cv       ; ci_waiting : cv.
```

- Segunda versión

Monitor

np: int = 0 | np_waiting: int = 0
ncd: int = 0 | ncd_waiting: int = 0
nci: int = 0 | nci_waiting: int = 0
nocoches: uc = True | p_waiting: uc = False
nop_ncd: uc = True | cd_waiting: uc = False
nop_nci: uc = True | ci_waiting: uc = False

Invariantes

$$\left\{ \begin{array}{l} np, ncd, nci, np\_waiting, ncd\_waiting, nci\_waiting \geq 0 \\ np > 0 \Rightarrow ncd = 0 \wedge nci = 0 \\ ncd > 0 \Rightarrow nci = 0 \end{array} \right\} \equiv INV$$

Métodos

```
enter_np()
  {INV}
  cd_waiting.wait(ncd_waiting == 0)
  ci_waiting.wait(nci_waiting == 0)
  {INV ∧ ncd_waiting == 0 ∧ nci_waiting == 0}
  np_waiting = np_waiting + 1
  {INV ∧ np_waiting > 0}
  nocoches.wait(ncd == 0 ∧ nci == 0)
  {INV ∧ ncd == 0 ∧ nci == 0}
  np = np + 1
  np_waiting = np_waiting - 1
  if np_waiting == 0:
      p_waiting.notify()
  {INV ∧ np_waiting == 0}
```

```
enter_ncd()
    {INV}
    p_waiting.wait(np_waiting == 0)
    ci_waiting.wait(nci_waiting == 0)
    {INV ∧ np_waiting == 0 ∧ nci_waiting == 0}
    ncd_waiting = ncd_waiting + 1
    {INV ∧ ncd_waiting > 0}
    nop_nci.wait(np == 0 ∧ nci == 0)
    {INV ∧ np == 0 ∧ nci == 0}
    ncd = ncd + 1
    ncd_waiting = ncd_waiting - 1
    if ncd_waiting == 0:
        cd_waiting.notify()
    {INV ∧ ncd_waiting == 0}.
```

```
enter_nci()
    {INV}
    p_waiting.wait(np_waiting == 0)
    cd_waiting.wait(ncd_waiting == 0)
    {INV ∧ np_waiting == 0 ∧ ncd_waiting == 0}
    nci_waiting = nci_waiting + 1
    {INV ∧ nci_waiting > 0}
    nop_ncd.wait(np == 0 ∧ ncd == 0)
    {INV ∧ np == 0 ∧ ncd == 0}
    nci = nci + 1
    nci_waiting = nci_waiting - 1
    if nci_waiting == 0:
        ci_waiting.notify()
    {INV ∧ nci_waiting == 0}.
```

```
leave_np()
    {INV ∧ np > 0}
    np = np - 1
    if np == 0:
        nop_ncd.notify()
        nop_nci.notify()
    {INV}.
```

```
leave_ncd()
{INV ∧ ncd>0}
ncd = ncd - 1
if ncd == 0:
    nocodes.notify()
    nop - ncd.notify()
{INV}
```

```
leave_nci()
{INV ∧ nci>0}
nci = nci - 1
if nci == 0:
    nocodes.notify()
    nop - nci.notify()
{INV}.
```

- Problema de inanición:

(i) Si hay solo peatones, ¿pueden entrar coches eventualmente?
Sí ya que si hay algún coche esperando, enter_np() no se ejecuta completamente. Y cuando np==0 (hip. justicia), no habrá inanición entre coches.
Ocurre lo mismo si hay peatones esperando y los coches turnándose
Luego no hay problema de inanición

(ii) Si los peatones y los coches en sentido derecha se turnan, los coches en sentido izquierda entrarán eventualmente porque nci_waiting>0 y no se podrá ejecutar enter_np() ni enter_ncd() completamente.
Ocurre análogamente si los peatones se turnan con los coches en sentido izquierda.
Luego no hay problema de inanición.

- Problema de deadlock.

Iniciamos con np_waiting = ncd_waiting = nci_waiting = 0.
Supongamos que quieren entrar peatones, entonces enter_np() se puede ejecutar hasta np_waiting += 1. Entonces enter_ncd() y enter_nci() no pueden empezar. Eventualmente los peatones entrarán en el puente y np_waiting llegará a 0, luego, por ejemplo, enter_ncd() podrá empezar y ejecutará ncd_waiting += 1 las veces que sea.
Para que puedan entrar los coches sentido derecha, los peatones anteriores deben haber dejado el puente, cosa que ocurre ya que leave_np() se ejecutará eventualmente. Así, ncd_waiting llegará a 0 y podrán empezar enter_nci() ó enter_np() sin problemas de inanición por lo anterior.

Por tanto, no se puede dar la situación en la que np_waiting > 0, nad_waiting > 0 y nci_waiting > 0 al mismo tiempo. Es decir, no hay problema de deadlock.