

**UNIVERSIDAD RAFAEL LANDÍVAR**

**FACULTAD DE INGENIERÍA**

**SISTEMAS OPERATIVOS**

**SECCIÓN 1 VESPERTINA**

**ING. JULIO REQUENA**

# **PETERSON Y DEKKER**

**Julio Anthony Engels Ruiz Coto 1284719**

**GUATEMALA DE LA ASUNCIÓN, MARZO 14 DE 2023**

**CAMPUS CENTRAL**

### EJEMPLO 1:

```
{ } settings.json  Untitled-1  class Peterson { Untitled-2  class Peterson { Untitled-3  import java.util.concurrent.atomic.AtomicInteger; Untitled-4
1  import java.util.concurrent.atomic.AtomicInteger;
2
3  class Peterson {
4      private AtomicInteger turn = new AtomicInteger();
5      private volatile boolean[] flag = new boolean[2];
6
7      public void lock(int i) {
8          int j = 1 - i;
9          flag[i] = true;
10         turn.set(j);
11         while (flag[j] && turn.get() == j) {
12             //esperar
13         }
14     }
15
16     public void unlock(int i) {
17         flag[i] = false;
18     }
19 }
```

En el presente ejemplo, la clase Peterson utiliza un objeto AtomicInteger para el turno en lugar de un entero regular. Además, las banderas se marcan como volátiles para garantizar que los cambios se propaguen a través de todos los hilos de ejecución.

### EJEMPLO 2:

```
{ } settings.json  Untitled-1  class Peterson { Untitled-2  class Peterson { Untitled-3
1  class Peterson {
2      private volatile boolean[] flag = new boolean[2];
3      private volatile int turn = 0;
4
5      public void lock(int i) {
6          int j = 1 - i;
7          flag[i] = true;
8          turn = j;
9          while (flag[j] && turn == j) {
10             //esperar
11         }
12     }
13
14     public void unlock(int i) {
15         flag[i] = false;
16     }
17 }
18 }
```

Las banderas y el turno se marcan como volátiles para garantizar que los cambios se propaguen a través de todos los hilos de ejecución.

### EJEMPLO 3:

```
{ } settings.json  Untitled-1  class Peterson {  Untitled-2  class Pete

1  class Peterson {
2      private boolean[] flag = new boolean[2];
3      private int turn = 0;
4
5      public void lock(int i) {
6          int j = 1 - i;
7          flag[i] = true;
8          turn = j;
9          while (flag[j] && turn == j) {}
10     }
11
12     public void unlock(int i) {
13         flag[i] = false;
14     }
15 }
16
```

El presente ejemplo, la clase Peterson contiene dos métodos: lock y unlock. El método lock toma un parámetro i que indica el índice del proceso que está intentando acceder a la sección crítica. El método establece la bandera del proceso en true, establece el turno en el otro proceso y luego entra en un bucle mientras el otro proceso tiene la bandera en true y es su turno. El método unlock simplemente establece la bandera del proceso en false.

### EJEMPLO 1:

```
{ } settings.json  Untitled-1  import java.util.concurrent.atomic.AtomicInteger  Untitled-2

1  import java.util.concurrent.atomic.AtomicInteger;
2
3  class Dekker {
4      private volatile boolean[] flag = new boolean[2];
5      private AtomicInteger turn = new AtomicInteger();
6
7      public void lock(int i) {
8          int j = 1 - i;
9          flag[i] = true;
10         turn.set(j);
11         while (flag[j] && turn.get() == j) {
12             //esperar
13         }
14     }
15
16     public void unlock(int i) {
17         flag[i] = false;
18         turn.set(1 - i);
19     }
20 }
21
```

El presente ejemplo, la clase Dekker utiliza un objeto AtomicInteger para el turno en lugar de un entero regular. Además, las banderas se marcan como volátiles para garantizar que los cambios se propaguen a través de todos los hilos de ejecución.

## EJEMPLO 2:

```
{ } settings.json  Untitled-1  import java.util.concurrent.atomic.Atomic  Untitled-2  class Dekker {  Untitled-3  1  class Dekker {  
2      private boolean[] flag = new boolean[2];  
3      private int turn = 0;  
4  
5      public void lock(int i) {  
6          int j = 1 - i;  
7          flag[i] = true;  
8          while (flag[j]) {  
9              if (turn == j) {  
10                 flag[i] = false;  
11                 while (turn == j) {}  
12                 flag[i] = true;  
13             }  
14         }  
15     }  
16  
17     public void unlock(int i) {  
18         turn = 1 - i;  
19         flag[i] = false;  
20     }  
21 }  
22
```

El presente ejemplo, la clase Dekker contiene dos métodos: lock y unlock. El método lock toma un parámetro i que indica el índice del proceso que está intentando acceder a la sección crítica. El método establece la bandera del proceso en true y entra en un bucle mientras el otro proceso tiene la bandera en true y es su turno. Si es el turno del otro proceso, entonces este proceso libera su bandera y espera a que el otro proceso termine. El método unlock establece el turno en el otro proceso y libera la bandera del proceso.

## EJEMPLO 3:

```
{ } settings.json  Untitled-1  import java.util.concurrent.atomic.Atomic  Untitled-2  class Dekker {  Untitled-3  class Dekker {  Untitled-4  1  class Dekker {  
2      private volatile boolean[] flag = new boolean[2];  
3      private volatile int turn = 0;  
4  
5      public void lock(int i) {  
6          int j = 1 - i;  
7          flag[i] = true;  
8          turn = j;  
9          while (flag[j] && turn == j) {  
10             //esperar  
11         }  
12     }  
13  
14     public void unlock(int i) {  
15         flag[i] = false;  
16         turn = 1 - i;  
17     }  
18 }  
19
```

En este ejemplo, la clase Dekker las banderas y el turno se marcan como volátiles para garantizar que los cambios se propaguen a través de todos los hilos de ejecución.