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:

```java
import java.util.concurrent.atomic.AtomicInteger;

class Peterson {
  private AtomicInteger turn = new AtomicInteger();
  private volatile boolean[] flag = new boolean[2];

  public void lock(int i) {
    int j = 1 - i;
    flag[i] = true;
    turn.set(j);
    while (flag[j] && turn.get() == j) {
      //esperar
    }
  }

  public void unlock(int i) {
    flag[i] = false;
  }
}
```

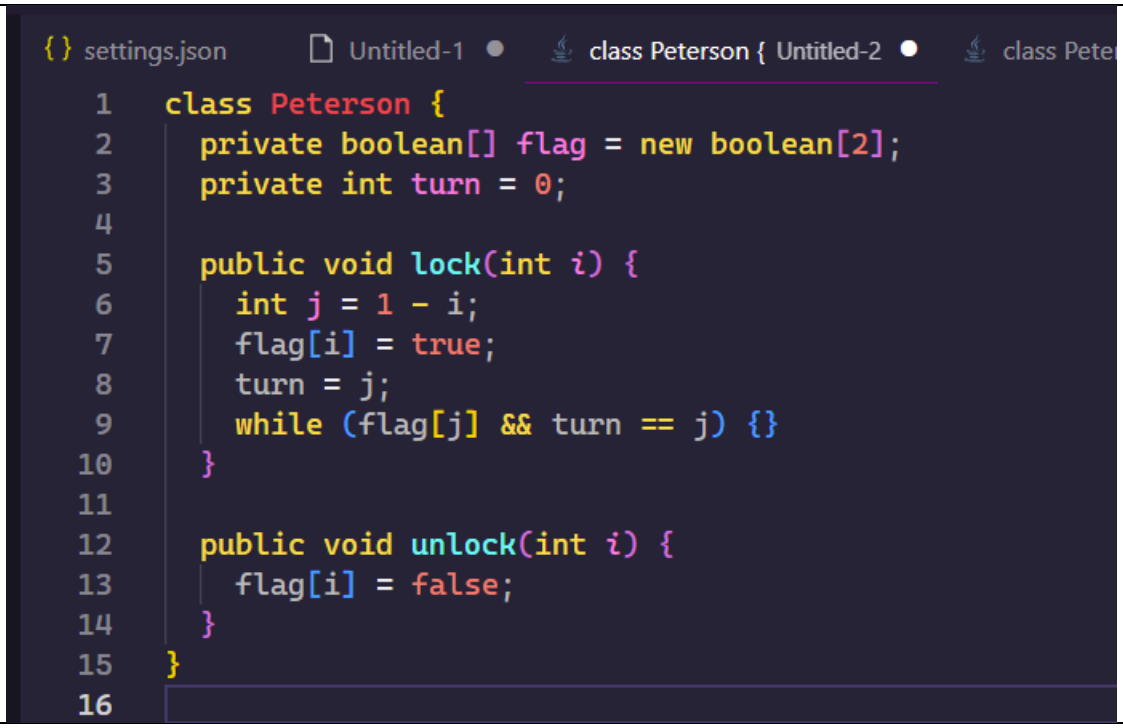
En el presente ejemplo, la clase Peterson utiliza un objeto AtomicInteger para el turno en lugar de un entero regular. Además, las banderas se marcan como volátiles para garantizar que los cambios se propaguen a través de todos los hilos de ejecución.

EJEMPLO 2:

```java
class Peterson {
  private volatile boolean[] flag = new boolean[2];
  private volatile int turn = 0;

  public void lock(int i) {
    int j = 1 - i;
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j) {
      //esperar
    }
  }

  public void unlock(int i) {
    flag[i] = false;
  }
}
```

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:



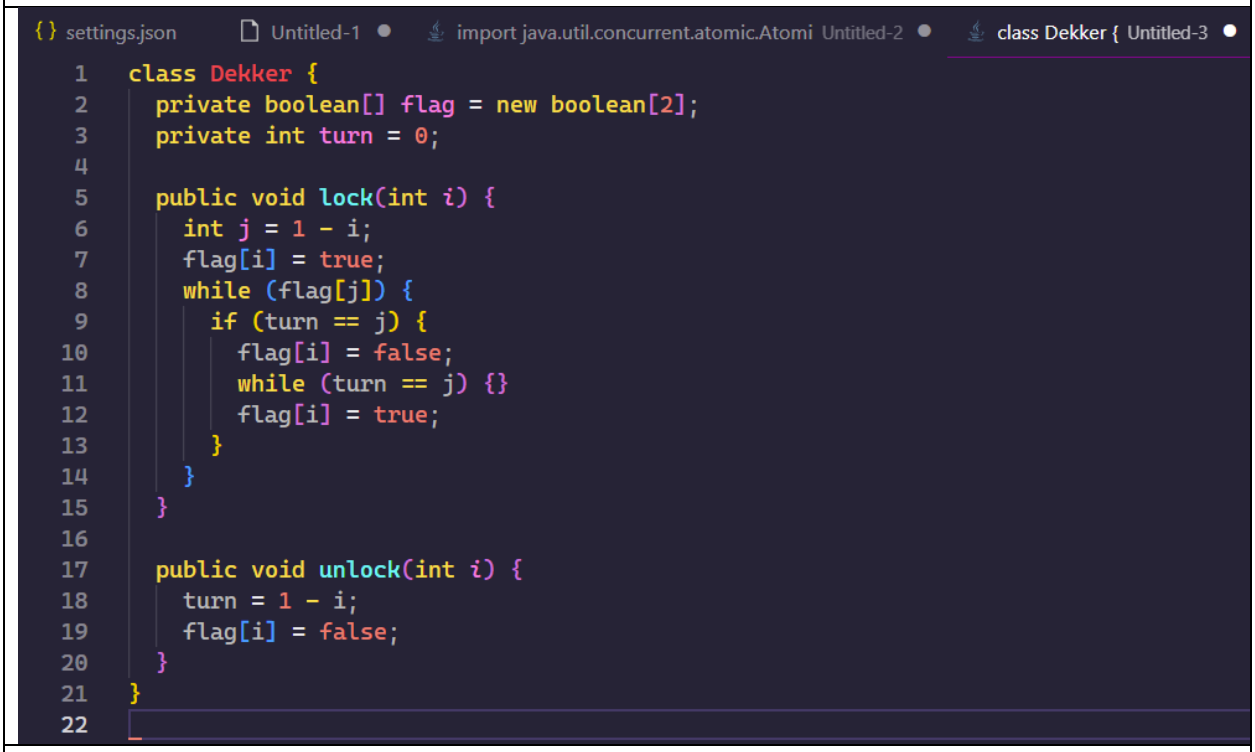
El presente ejemplo, la clase Peterson contiene dos métodos: lock y unlock. El método lock toma un parámetro i que indica el índice del proceso que está intentando acceder a la sección crítica. El método establece la bandera del proceso en true, establece el turno en el otro proceso y luego entra en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


```java
import java.util.concurrent.atomic.AtomicInteger;

class Dekker {
  private volatile boolean[] flag = new boolean[2];
  private AtomicInteger turn = new AtomicInteger();

  public void lock(int i) {
    int j = 1 - i;
    flag[i] = true;
    turn.set(j);
    while (flag[j] && turn.get() == j) {
      //esperar
    }
  }

  public void unlock(int i) {
    flag[i] = false;
    turn.set(1 - i);
  }
}
```

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.

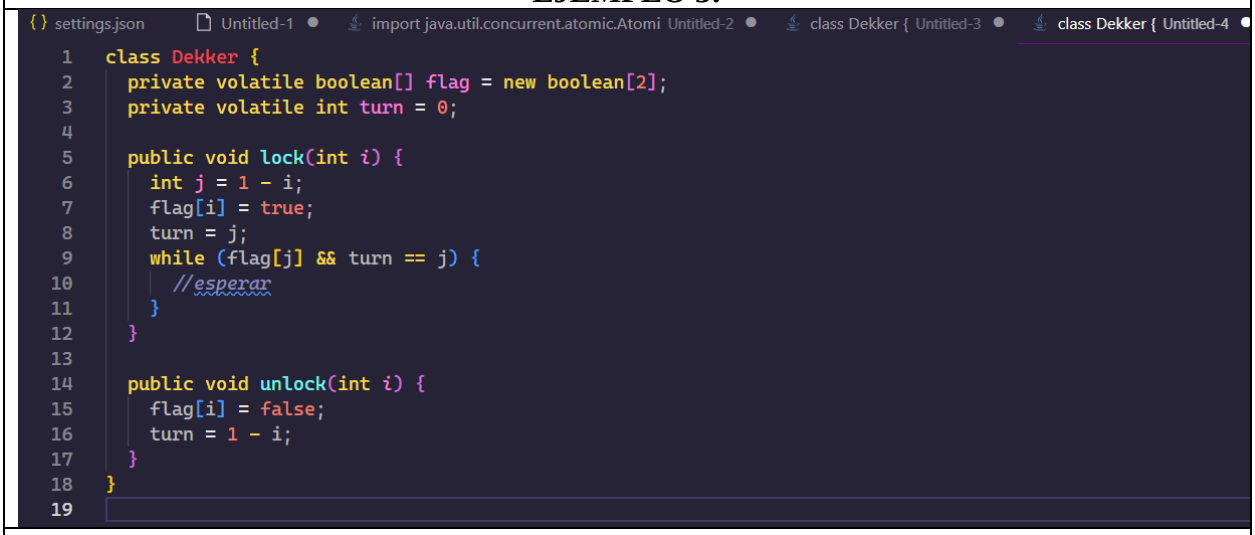
EJEMPLO 2:




```java
class Dekker {
  private boolean[] flag = new boolean[2];
  private int turn = 0;

  public void lock(int i) {
    int j = 1 - i;
    flag[i] = true;
    while (flag[j]) {
      if (turn == j) {
        flag[i] = false;
        while (turn == j) {}
        flag[i] = true;
      }
    }
  }

  public void unlock(int i) {
    turn = 1 - i;
    flag[i] = false;
  }
}
```

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:




```java
class Dekker {
  private volatile boolean[] flag = new boolean[2];
  private volatile int turn = 0;

  public void lock(int i) {
    int j = 1 - i;
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j) {
      //esperar
    }
  }

  public void unlock(int i) {
    flag[i] = false;
    turn = 1 - i;
  }
}
```

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.