Locks explícitos e Variáveis Condicionais

O exemplo Produtor/Consumidor

- Um produtor gera um número entre 0 e 9 (a variável i) e o armazena num objeto chamado CubbyHole **(um lugarzinho para guardar coisas; um cubículo)**
- Para deixar o exemplo mais interessante, o produtor dorme durante um intervalo aleatório entre 0 1 100 ms antes de gerar mais números
    - No programa abaixo, "number" é a identificação do produtor, "i" é o número gerado.

```
public class Producer extends Thread {
    private CubbyHole cubbyhole;
    private int number;

    public Producer(CubbyHole c, int number) {
        cubbyhole = c;
        this.number = number;
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            cubbyhole.put(number, i);
            try {
                sleep((int)(Math.random() * 100));
            } catch (InterruptedException e) { }
        }
    }
}
```

- O consumidor consome os inteiros do mesmo CubbyHole asssim que se tornam disponíveis

```
public class Consumer extends Thread {
    private CubbyHole cubbyhole;
    private int number;

    public Consumer(CubbyHole c, int number) {
        cubbyhole = c;
        this.number = number;
    }

    public void run() {
        int value = 0;
        for (int i = 0; i < 10; i++) {
            value = cubbyhole.get(number);
        }
    }
}
```

- Para proteger regiões críticas (proteger uma seção de código), podemos também usar um lock explícito.
    - Um lock explícito é mais flexível do que usar a palavra synchronized:
        - Permite proteger alguns statements (sem bloco)
        - permite proteger múltiplos métodos
- Para criar um lock explícito, você instancia uma implementação da interface Lock
    - Normalmente, instancia-se ReentrantLock
- Para obter o lock, usa-se o método lock()
- Para liberar o lock: usa-se unlock()
- Já que o lock não é liberado automaticamente no final de um método, pode ser útil usar try/finally para garantir que o lock seja liberado
- Para esperar por um lock explícito, cria-se uma variável condicional
    - Um objeto que implementa a interface Condition
    - Usar Lock.newCondition() para criar uma condição
- A condição provê métodos:
    - "await" para esperar até a condição ser verdadeira.
    - "signal" e "signalAll" para avisar os threads que a condição ocorreu.
    - As variantes de "await" aparecem na tabela seguinte:

Métodos Condition.await	
Método	**Descrição**
`await`	Espera uma condição ocorrer
`awaitUninterruptibly`	Espera uma condição ocorrer. Não pode ser interrompido.
`awaitNanos(long timeout)`	Espera uma condição ocorrer. Espera no máximo timeout nanossegundos
`await(long timeout, TimeUnit unit)`	Espera uma condição ocorrer. Espera no máximo timeout TimeUnit
`await(Date timeout)`	Espera uma condição ocorrer. Espera no máximo até a data especificada

- O exemplo abaixo é `CubbyHole` reescrito para usar um **lock explícito** e uma **variável condicional.**

```
import java.util.concurrent.locks.*;
public class CubbyHole2 {
    private int contents;
    private boolean available = false;
    private Lock aLock = new ReentrantLock();
    private Condition condVar = aLock.newCondition();

    public int get(int who) {
        aLock.lock();
        try {
            while (available == false) {
                try {
                    condVar.await();
```

```
                } catch (InterruptedException e) { }
            }
            available = false;
            System.out.println("Consumer " + who + "
got: " + contents);
            condVar.signalAll();
        } finally {
            aLock.unlock();
            return contents;
        }
    }

    public void put(int who, int value) {
                    aLock.lock();
    try {
        while (available == true) {
            try {
                condVar.await();
            } catch (InterruptedException e) { }
        }
        contents = value;
        available = true;
        System.out.println("Producer " + who + " put:
" +
                                contents);
        condVar.signalAll();
        } finally {
            aLock.unlock();
        }
    }
}
```

- Para rodar, execute ProducerConsumerTest2

```
public class ProducerConsumerTest2 {
    public static void main(String[] args) {
        CubbyHole2 c = new CubbyHole2();
        Producer2 p1 = new Producer2(c, 1);
        Consumer2 c1 = new Consumer2(c, 1);
        p1.start();
        c1.start();
    }

}
```