Relatório EP 3 · SO

Prof. Dra. Gisele da Silva Craveiro

Mikael Gi Sung Shin, 10843441

Código e Explicação:

```java
public class Main {

    public static void main(String[] args) {

        MyThread[] thread = new MyThread[2];

        for(int i = 0; i <= 1; i++){

            thread[i] = new MyThread("T" + i);
        }
    }
}
```

1) Este é o método main(). Ele apenas instancia dois threads da classe MyThread, que herda a classe Thread existente no próprio Java, para propor uma solução ao exercício programa.

```java
import java.util.concurrent.Semaphore;

public class MyThread extends Thread {

    public static int loop = 1000000000;
    public static Semaphore mutex = new Semaphore(1);
    public String name; // atributo nome da thread

    public MyThread(String name){

        this.name = name;
        start();
    }
```

2) Descrição da imagem acima:

- faz o import da biblioteca Semaphore;
- define a variável loop usada como um limite;
- instancia o objeto “mutex”, que será o nosso Semáforo binário;
- declara o atributo nome;
- o construtor dessa classe recebe o nome, e em seguida, é posto para sua execução com o método start().

```
public void run(){

    System.out.println(this.name + " chegou no run().");

    try {
        while(true){
            while(mutex.availablePermits() == 0){
                Thread.sleep(1);
            }
            if(mutex.availablePermits() == 1){
                mutex.acquire();
                secaoCritica();
            }
            secaoNaoCritica();
            Thread.sleep(1000);
        }
    } catch(Exception e) {

    }
}
```

3) Primeiro, é exibida a mensagem de sucesso para a thread ‘n’ (como consta na saída do programa na página 3), que conseguiu ser colocada para execução. No bloco try, contém um loop infinito, e dentro dele, há um outro que fica em constantes consultas a fim de verificar se o semáforo é igual a 0, isto é, se não há nenhum “recurso” disponível para a alocação do processo. Enquanto não tiver nenhum recurso, o processo é posto para dormir. Agora, quando existe o recurso (semáforo = 1), o processo entra na seção crítica e o valor do semáforo é “setado” para 0, pois este último processo está a ocupando, e nenhum outro poderá entrar até que o semáforo volte a ser 1. Por fim, o processo executa um método simulando que não pertence a região crítica e, em seguida, é posto para dormir.

```
public void secaoCritica(){

    System.out.println(this.name + ": ENTRANDO na Seção Crítica...");

    try {

        System.out.println("Semáforo: " + mutex.availablePermits());
        for (int i = 0; i < loop; i++)
            i = i;
        mutex.release();
        System.out.println(this.name + ": SAINDO da Seção Crítica...");
        System.out.println("Semáforo: " + mutex.availablePermits());
    } catch(Exception e) {

    }
}
```

4) Descrição das linhas de código acima:

- exibe uma mensagem que tal thread conseguiu entrar nessa seção;
- imprime o ‘status’ do mutex;

- executa um for qualquer;
- incrementa o mutex para disponibilizar a seção crítica/recurso e imprime.

```
    public void secaoNaoCritica(){

        for (int i = 0; i < loop; i++)
            i = i;
        System.out.println("---------------------------------------");
    }
}
```

5) Método que simula a execução de uma região não crítica.

Sobre os requisitos para uma boa solução da região crítica, esta implementação obedece, é claro, a Exclusão Mútua, pois se o 'processo 0' executa a região crítica, então, nenhum outro entra nesta região. Porém, desrespeita as outras duas, a de Espera Limitada, pois não há nenhum limite para quantas vezes um processo entra nesta seção, e também, a de Progresso, pois o algoritmo não define uma ordem nos processos que irão executar tal região posterior ao processo que já se encontra nela.

Saída:

```
T0: ENTRANDO na Seção Crítica...
Semáforo: 0
T0: SAINDO da Seção Crítica...
Semáforo: 1
---------------------------------------
T1: ENTRANDO na Seção Crítica...
Semáforo: 0
T1: SAINDO da Seção Crítica...
Semáforo: 1
---------------------------------------
T0: ENTRANDO na Seção Crítica...
Semáforo: 0
T0: SAINDO da Seção Crítica...
Semáforo: 1
---------------------------------------
T1: ENTRANDO na Seção Crítica...
Semáforo: 0
T1: SAINDO da Seção Crítica...
Semáforo: 1
---------------------------------------
```

Instruções para compilação:

1) Usar alguma IDE que suporte a linguagem java, e execute ("run") o código descrito acima. (obs: utilizei o compilador online https://www.onlinegdb.com/online_c_compiler).
2) Executar as seguintes linhas de código em algum terminal, dentro do diretório do programa:
   `javac "nomedoarquivo".java`
   `java "nomedoarquivo"`

Referências para o desenvolvimento do código:

1) https://www.youtube.com/watch?v=LZM-9A8jWeg&t=752s&ab_channel=UNIVESP
2) https://www.youtube.com/watch?v=JaXr15a5cPA&ab_channel=UNIVESP
3) https://www.youtube.com/watch?v=KxTRsvgqoVQ&t=321s&ab_channel=CaveofProgramming
4) Fundamentos de Sistemas Operacionais de Abraham Silberschatz.