Sistemas de Computação

Pré-requisitos?

Para execução do código, é exigido o JDK devidamente configurado. O *Java Development Kit* é uma implementação das plataformas Plataforma Java, Standard Edition, Plataforma Java, Enterprise Edition ou Plataforma Java, Micro Edition lançadas pela Oracle Corporation na forma de um produto binário destinado a desenvolvedores Java no Solaris, Linux, macOS ou Windows.

Como executar o código?

Comando de compilação:

```
java -jar "Trabalho2_Semaforo Threading_SO.jar"
```

Semáforo e Threads

Neste trabalho, o desenvolvimento de semáforos e threads foram realizados utilizando a linguagem de programação Java, pois fornece diversas APIs para implementação de softwares utilizando o paralelismo de tarefas computacionais. Na programação em multithreading, os objetos de thread possuem identificador único, incluindo espaços de armazenamento utilizadas pelo JVM e pelo sistema operacional, salvando assim seu contexto enquanto permanece em modo de espera pelo escalonador.

No trabalho em questão, é necessário imprimir três cores (vermelho, azul e verde) por n vezes e nesta mesma ordem. Para esta implementação, foram construídos três threads que são responsáveis por apenas uma das cores da lista. Os nomes atribuídos para cada foram, a thread A é responsável pela cor vermelha, thread B pela cor verde e a thread C pela cor azul. Os threads são implementados em classes e funções diferentes e unidas em uma classe única. Além disso, cada thread possui um semáforo responsável pelo controle de recursos e liberação de execução, como cada thread foi implementada separadamente, então há um semáforo para cada thread.

Cada thread foi implementado de maneira semelhante, já que elas são responsáveis basicamente por executar uma mesma tarefa (imprimir uma cor específica na tela). Com isso, cada *thread* possui o seu próprio semáforo e cada um foi inicializado com um valor inicial diferente A Tabela 3 indica a estrutura principal da classe responsável por todo processo. Há três semáforos, estes correspondem a cada *thread* {Red: ThreadA, Blue: ThreadB e Green: ThreadC}. A especificação das funções pode ser vista nos tópicos a seguir:

- A função timeSleeping() é responsável por selecionar um valor aleatório para que a thread correspondente entre em estado de sleeping. Isso quer dizer que a thread deve esperar o tempo retornado por este método, que varia de 0 a 9 segundos.

- A função RecursoImprimir() é responsável pela impressão da cor. Esta função foi desenvolvida para simular um único recurso em que as threads precisam acessar por vez.
- A função Red() é responsável pela requisição da ThreadA. Nela, há uma função Red.require(), que concede a execução apenas da ThreadA para o recurso de impressão. Após sua execução, o recurso precisa liberado para a próxima thread da lista, o que é feito por meio do comando release().
- A função Blue() é responsável pela requisição da ThreadB. Nela, há uma função Blue.require(), que concede a execução apenas da ThreadA e libera o recurso em sua finalização para a ThreadC por meio do comando Green.release().
- Por fim, a função Green() é responsável unicamente pela requisição da ThreadC. Ao finalizar o uso do recurso, esta chama novamente a ThreadA (lista circular), caso ainda haja execuções pendente.

Dessa maneira, cada thread tem um semáforo responsável pela sua execução ou não. A Figura 1 exemplifica como cada semáforo foi inicializado. Como a thread A é a primeira a executar, então esta foi atribuída com o valor 1, enquanto o restante dos threads tem o valor de semáforo igual a 0.

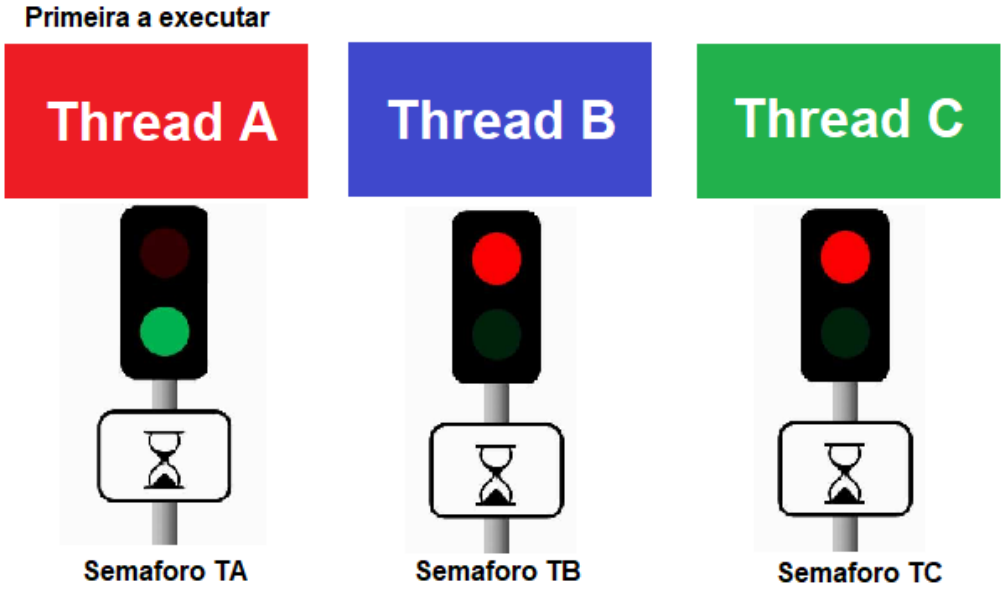

Figura 1 - Inicialização das *Thread*s e Semáforos

Já na execução, como temos um recurso concorrente (função de imprimir), somente um dos *thread*s utilizará o recurso, bloqueando-o para as demais. Logo, quando o valor de semáforo de uma delas é v = 1 (indica que está em execução ou liberado), as demais estão com o valor v = 0 (em espera para execução, recurso em uso por outro *thread*). Além disso, o código foi desenvolvido para que caso a *thread* esteja com status finalizado, esta libera o recurso para a próxima *thread* da pilha, respeitando a ordem pré-estabelecida de impressão das cores.

Interface de Execução

A interface foi desenvolvida para exibição de códigos diretamente por linha de comando (prompt de comando). Como exemplo, considere a Figura 3, esta demonstra o que significa cada um dos campos exibidos ao executar o código em Java. Cada item está definido conforme a lista abaixo:

- Item (a): Indica qual o nome do thread que o Semáforo permitiu a execução
- Item (b): Indica o tempo que um thread entrou em modo sleeping conforme uma função randômica, o tempo de sleeping varia de 0 a 9 segundos para cada thread. Além disso, a impressão só é realizada após este tempo de sleeping do thread.
- Item (c): Indica a impressão (vermelho, azul ou verde), conforme a thread em execução.
- Item (d): Indica o instante em que o recurso foi liberado (Recurso de imprimir na tela).

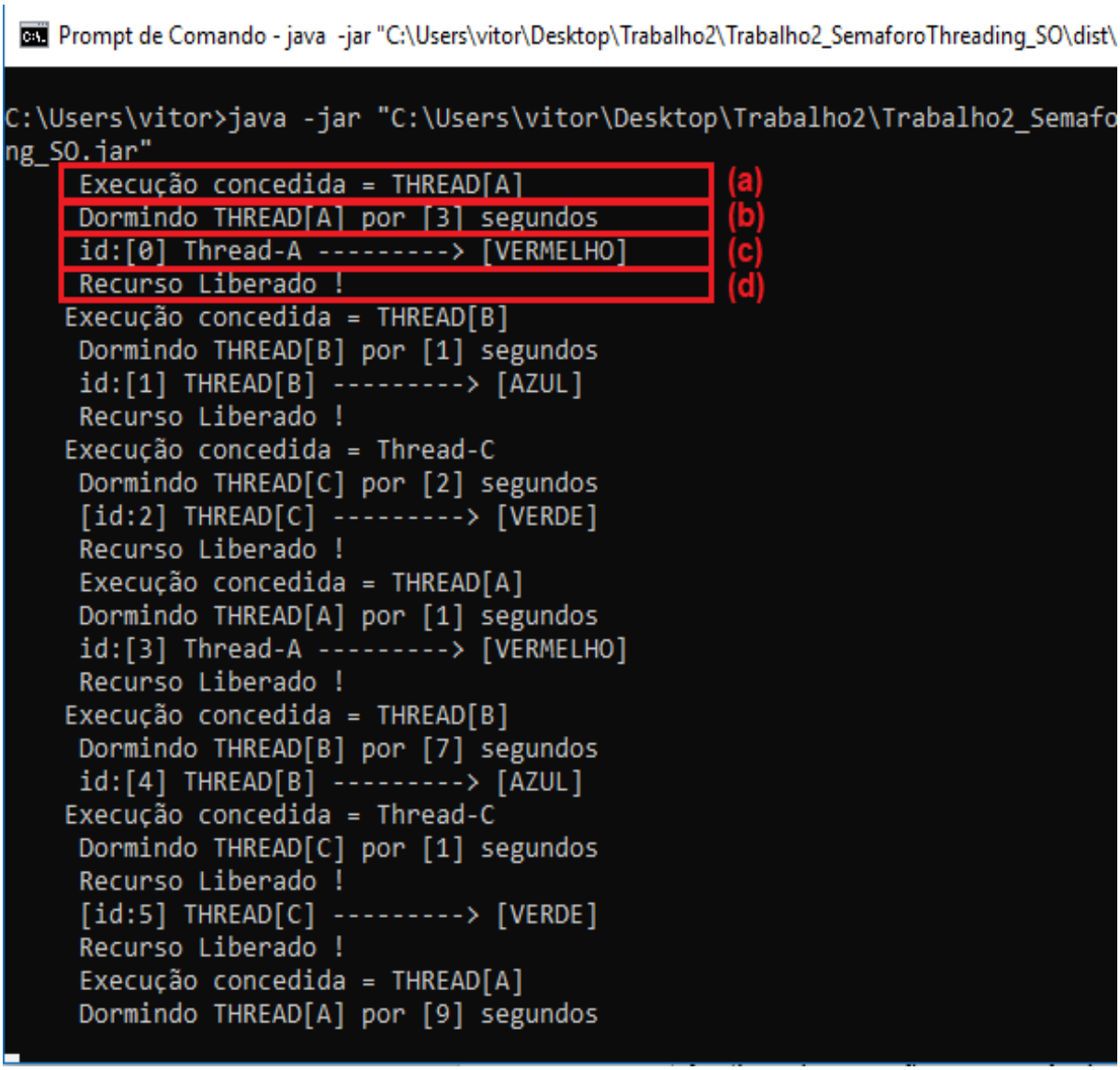

Figura 2 - Interface de Execução