PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO – 2018-1

Profa. Heloisa – Terceiro Trabalho – Programação Concorrente

==========

Escreva um programa em JAVA para simular um conjunto de carros estacionando e saindo de um estacionamento. O estacionamento deve possuir pelo menos 5 vagas, que são representadas em um objeto compartilhado pelas linhas de execução que são os carros tentando estacionar (pelo menos 8).

- Utilizar uma classe Estacionamento, com os campos de dados que representam as vagas e com os métodos estacionar (...) e sair(...).
- As vagas podem estar em dois estados: livre ou ocupada.
- O método estacionar(...) procura uma vaga livre (sequencialmente) e, se encontra, estaciona. Se não, espera até liberar uma vaga.
- A operação de estacionar consiste em alterar o estado da vaga sendo utilizada de livre para ocupada.
- O método sair(...) sai do estacionamento e libera a vaga que estava sendo utilizada pelo carro.
- A operação de sair consiste em alterar o estado da vaga sendo liberada de ocupada para livre.
- Sincronizar os métodos estacionar(...) e sair(...) para garantir que essas operações não serão feitas simultaneamente por mais de uma linha de execução.
- Usar os métodos wait() e notify(), dentro do método estacionar(...), para fazer o sincronismo de cooperação, de tal forma que um carro estacione somente quando houver uma vaga livre e avise que concluiu sua tarefa, liberando o estacionamento, depois de estacionar.
- Usar o método notify( ) dentro do método sair(...) para fazer o sincronismo de cooperação, avisando que saiu e liberou o estacionamento.
- Utilizar uma classe EstacionaThread que inicia uma linha de execução para cada carro. Seu método run deve chamar o método estacionar(...), esperar algum tempo e chamar o método sair(...).
- O tempo que o carro fica estacionado deve ser gerado aleatoriamente e passado para o método sleep.
- A cada operação de estacionar, imprimir uma mensagem indicando qual carro está estacionando e qual vaga está sendo ocupada. Ao liberar a vaga, imprimir uma mensagem dizendo qual vaga está sendo liberada e qual carro está saindo da vaga.
- Cada carro deve estacionar e sair um número de vezes entre 2 e 4, gerado aleatoriamente para cada um podendo, portanto, ser diferente para cada carro.
- O programa pode parar depois que todos os carros tiverem realizado o número de estacionamentos determinado para cada um. O controle do término de execução de cada Thread (carro) pode ser feito com o comando for no método run ou usando os métodos interrupt() e interrupted().
- Exemplo da classe EstacionaThread:

```
class EstacionaThread extends Thread  {
   private Estacionamento park; // Objeto compartilhado pelas threads
   private int quantasvezes; // Número vezes que o carro deve estacionar e sair
  public EstacionaThread (Estacionamento e, int qv) {
      park = e;
      quantasvezes = qv; /
   }

  public void run ( ) {
    try {
         while (!interrupted ( )) {
            park .estacionar ( );
            // Pode imprimir aqui
          int valor = (int) (Math.random( ) * 1000);
            sleep(valor);
            park.sair();
            // Pode imprimir aqui
         }
    } catch (InterruptedException e) {}
  }
}
```

Observações:

- Os trabalhos podem ser feitos em duplas, sendo as mesmas dos trabalhos anteriores;
- Entregar, no ambiente da disciplina no Moodle-DC, listagem do programa (arquivo .java), programa compilado (arquivo .class), relatório com documentação do programa (explicação do funcionamento de cada classe) e exemplos de execução.
- Data de entrega: 17/07/2018.