Exercicio 3.1

Experimente trocar, no exemplo mostrado, os dois `notifyAll()` por `notify()`, e tente perceber se a performance fica melhor ou pior. Veja se a aplicação ainda funciona como esperado (execute algumas vezes.) Embora `notify()` não acorde todos os threads, ele poderá liberar a trava para um thread consumidor quando não há nada para consumir, ou para um produtor quando não houver espaço para produzir, retendo a trava em cada operação.

Exercicio 3.2

Escreva uma classe `Pilha` com métodos sincronizados `char pop()` e `push(char)` que armazenem caracteres em um buffer de seis posições (use um array de 6 elementos) e um cursor (inteiro) que aponte para o próximo espaço livre. O método `push()` deve inserir um caractere no array (se houver espaço) e incrementar a posição do cursor. Se não houver espaço, o thread deve esperar até que um espaço seja liberado (essa é a condição). Tendo inserido um caractere, ele deve notificar outros threads que queiram consumi-lo. O método `pop()` deve devolver o último caractere inserido (se houver caracteres) e decrementar a posição do cursor. Se não houver caracteres, ele deve esperar até que um seja inserido. Tendo consumido um caractere, ele deve notificar outros threads que queiram inserir.

a. Escreva um produtor que chame `push(char)` 26 vezes, enviando as 26 letras do alfabeto para a `Pilha`.
b. Escreva um `Consumidor` que chame `pop()` dentro de um loop infinito e imprima cada caractere encontrado.
c. Escreva uma classe executável que crie uma pilha, um thread consumidor e um thread produtor e teste a aplicação.