INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE
Campus Natal - Central

Exercícios de Estrutura de Dados - Pilha

1. Descreva o estado da pilha (inicialmente vazia) após cada uma das operações a seguir (Esta questão é a mesma da R-4.1 do livro).
   push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop().

2. Utilizando os arquivos PilhaArray.java, Pilha.java, TestePilha.java (http://docente.ifrn.edu.br/robinsonalves), preencha a tabela abaixo, onde a linha superior diz o método de aumento da capacidade e a coluna da esquerda informa a quantidade de elementos colocados na pilha. Em cada célula da tabela deverá ser colocado o tempo, em milisegundos (use System.currentTimeMillis()), de se colocar os elementos. Utilize as duas estratégias de crescimento (constante e duplicação) para o *array*.

	incremento 10	incremento 100	incremento 1000	duplicação
10 elementos				
100 elementos				
1000 elementos				
10000 elementos				
100000 elementos				
1000000 elementos				

   Sugestão: Tente implementar a interface Pilha.java e use o arquivo PilhaArray.java só em último caso para tirar dúvidas.

3. Justifique os valores obtidos no exercício anterior.

4. Seja a método empty( ) tal que esvazie a pilha descartando todos os seus elementos. acrescente o método empty( ) a classe PilhaArray..

5. Projete uma classe que contenha duas pilhas, uma "vermelha" e outra "preta" e suas operações são adaptações "coloridas" das operações habituais sobre pilhas. Por exemplo, esta classe deve prover uma operação de *push* vermelha e uma operação de *push* preta. Usando um único array cuja a capacidade é limitada por um tamanho *N* que é sempre maior do que os tamanhos somados das duas pilhas. (P-42). A pilha "vermelha" pode começar no início do *array* e a pilha "preta" pode começar no final do *array*.

6. Implemente o TAD pilha usando a classe Vector embutida em Java (P-43). Use a interface pilha (Pilha.java – http://docente.ifrn.edu.br/robinsonalves).

7. Crie uma classe que implemente a interface PilhaInt.java (http://docente.ifrn.edu.br/robinsonalves) Acrescente um método chamado adicionaPilha que receba como parâmetro uma pilha e coloque todos os elementos desta pilha no topo da pilha que chamou o método. A assinatura do método segue abaixo.

   public void adicionaPilha(Pilha p);

Exemplo: se uma pilha P1 continha 5 elementos e o método adicionaPilha foi chamado e tinha como parâmetros uma pilha P2 com 5 elementos, a pilha P1 passará a ter 10 elementos, onde os 5 inferiores são os 5 que ela já tinha e os 5 superiores são os que estavam na pilha P2. A pilha P2 não se altera.

8. Em uma expressa aritmética os símbolos de parentização parênteses (()), colchetes ([]) e chaves ({}) devem ser balanceados e apropriadamente aninhados. Apresente um algoritmo, em pseudo-código, que verifica se uma determinada expressão aritmética é correta ou não. O algoritmo deve executar em tempo O(n), onde n é a quantidade de caracteres da expressão e deve usar uma pilha como estrutura de dados auxiliar (C-4.6).