

**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA- UESB**

**DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS- DCET**

**DOCUMENTAÇÃO DO PROJETO REFERENTE ÀS ESTRUTURAS DE DADOS – PILHA, LISTA E FILA**

**BRUNNA REIS SOUZA  
NYLO FIGUEIRA PINTO**Professor: Marcos Prado

VITÓRIA DA CONQUISTA – BAHIA – BRASIL

SETEMBRO – 2016

**Introdução**

O projeto consiste em um programa que possibilitará o estudo das estruturas de dados Pilha, Lista e Fila, com o objetivo de que os usuários consigam manipulá-las de acordo com suas características principais. Uma pilha é uma estrutura de dados que admite remoção de elementos e inserção de novos objetos. Mais especificamente, uma pilha (= stack) é uma estrutura sujeita à seguinte regra de operação: sempre que houver uma remoção, o elemento removido é o que está na estrutura há menos tempo. Em outras palavras, o primeiro objeto a ser inserido na pilha é o último a ser removido (LIFO = Last in – First Out).   
As filas são estruturas baseadas no princípio FIFO (first in, first out), em que os elementos que foram inseridos no início são os primeiros a serem removidos. Uma fila possui duas funções básicas, enqueue, que adiciona um elemento ao final da fila, e dequeue que remove o elemento no início da fila.  
Uma lista implementa uma coleção ordenada de valores onde o mesmo valor pode ocorrer mais de uma vez. A estrutura de lista estática permite apenas a verificação e enumeração dos valores, já uma lista dinâmica, pode permitir que itens sejam inseridos, substituídos ou excluídos durante a existência da lista. Além dos botões de inserção e remoção utilizados na classe lista, teremos também três funções (Insertion sort, selection sort e bubble sort) para ordenar a lista de forma crescente. Insertion sort, ou ordenação por inserção, é um simples algoritmo de ordenação, eficiente quando aplicado a um pequeno número de elementos. Em termos gerais, ele percorre um vetor de elementos da esquerda para a direita e à medida que avança vai deixando os elementos mais à esquerda, ordenados. A ordenação por seleção (do inglês, *selection sort*) é um algoritmo de ordenação baseado em se passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os {\displaystyle n-1} elementos restantes, até os últimos dois elementos. Por fim, o Bubble sort, ou ordenação por flutuação (literalmente "*por bolha*"), é um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, a cada passagem fazendo flutuar para o topo o maior elemento da sequência. O código foi implementado através da linguagem de programação Java e sua interface através da plataforma de software multimídia JavaFX.

**Descrição geral do problema**

As três estruturas utilizadas são dinâmicas. No entanto, para representarmos de forma objetiva na interface, definimos um tamanho fixo de 10 elementos por estrutura. Assim, quando o usuário tentar inserir mais de 10 elementos, aparecerá uma mensagem informando que a estrutura x está cheia. A partir desse ponto, o usuário decide se remove algum elemento para inserir outro ou reseta a estrutura.

**Classes utilizadas**

**No:** nesta classe foram declaradas as variáveis “info” e “proximo”. O papel desta classe é basicamente manter referências dos outros objetos.

**Pilha:** nesta classe foi declarada a variável “topo”, que recebe a informação do elemento que está no topo da pilha, e os principais métodos que definem essa estrutura. O método “isEmpty” verifica se a pilha está vazia. O “push” empilha/insere elementos na pilha e o “pop” desempilha/remove elementos da pilha. Foi criado também o método “pilhaParaVetor”, que recebe um vetor, um objeto e um número inteiro que determinará a posição do vetor e retornar uma pilha.

**Fila:** nesta classe foram declaradas as variáveis “ultimo” e “primeiro”. O método “isEmpty” verifica se a fila está vazia. O método “enQueue” insere elementos na fila e o “deQueue” remove elementos. O método “pilhaParaVetor” recebe um vetor, um objeto e um número inteiro que determinará a posição do vetor e retornar uma fila.

**Lista:** nesta classe foram declaradas as variáveis “inicio”, que armazena o primeiro elemento da lista e “numPos”, que informa a posição do elemento na lista. Diferente da pilha e da fila, a lista pode ser manipulada de formas diferentes. O método “incluirInicio”, insere elementos no início da lista. O método “removerElemento”, pode remover um elemento x em qualquer posição da lista. O método “removerPosicao”, pode remover qualquer posição da lista. O “incluirPosicao” permite que o usuário insira um elemento em qualquer posição da lista. Os métodos “trocaElementoPos, troca e buscaPos” são utilizados nos três métodos de ordenação “insetionSort, selectionSort e bubble sort”.

**ArvoreFXProject**: Classe principal do programa. Aqui é setado onde está e como se chama o arquivo FXML, que por sua vez contém todas as informações XML usadas pelo JavaFX. Aqui foi criada a Scene, janela da GUI, e setado seu título.

**Controller**: Nessa classe serão setados todas as ações (e suas lógicas) que serão tomadas dentro do contexto GUI JavaFX.

Foram criados diversos vetores e objetos para manipulação e visualização na GUI.

* Métodos da pilha: o “adicionarNoPilha”, adicionará os nós na interface, através do método push da classe Pilha. Dentro desse método temos um “for” que verifica se determinada posição do vetor é diferente de nula, setando os retângulos e labels como visíveis.   
  O “removeNoPilha”, removerá os elementos do topo da pilha chamando o método “pop” da classe Pilha, de acordo com a lógica dessa estrutura.  
  O “resetPilha” limpará todos os elementos da tela, transformando todas as posições em nulas.
* Métodos da fila: “adicionarNoFila” adicionará os nós na interface, através do método “enqueue” da classe Fila. Dentro desse método temos um “for” que verifica se determinada posição do vetor é diferente de nula, setando os retângulos e labels como visíveis.  
  O “removeNoFila”, removerá os elementos do início da fila chamando o método “dequeue” da classe Fila, de acordo com a lógica dessa estrutura.  
  O “resetFila” limpará todos os elementos da tela, transformando todas as posições em nulas.
* Métodos da Lista: “adicionarInicioLista” adicionará elementos no início da lista, através da chamada do método “incluirInicio” da classe Lista.  
  O “adicionarPosicaoLista” adicionará elementos em qualquer posição da lista, através da chamada do método incluirPosicao da classe Lista.  
  O “removerElementoLista” removerá um elemento através da chamada do método removerElemento da classe Lista.  
  O “removerPosLista” removerá uma posição da lista através da chamada do método removerPosicao da classe Lista.  
  O “resetLista” limpará todos os elementos da tela, transformando todas as posições em nulas.  
  Temos ainda os métodos de ordenação “Insertion sort”, “selection sort” e “bubble sort” que ordenarão a lista de acordo com suas respectivas lógicas.

**Arquivo:** teste.fxml, fonte fxml onde é definida a GUI.

**Conclusão**Com este trabalho, foi possível fixar e entender melhor o conteúdo visto nesse semestre em sala de aula e revisar pontos-chave de cada estrutura. Assim, podemos compreender o uso e aplicação e cada uma, de forma que fique claro saber em que momento usá-las de acordo com o problema requerido, para assim otimizar o tempo de execução do código.

**Referências**

LAFORE, Robert. *Estruturas de Dados e Algoritmos em Java.* 2 ed. Rio de Janeiro: Editora Ciência Moderna Ltda., 2004.

PREISS, Bruno R. *Estruturas de Dados e Algoritmos: Padrões de projetos orientados a objeto com Java*. Rio de Janeiro: Editora Campus, 2000.

ARTIFICIAL INTELLIGENCE: A MODERN APPROACH. Stuart Russel, Prentice-Hall, 2nd Ed., 2002. [Base]

[T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein](http://en.wikipedia.org/wiki/Introduction_to_Algorithms),  [Introduction to Algorithms, 3rd edition](http://mitpress.mit.edu/books/introduction-algorithms),  MIT Press, 2009.   [Há uma edição em português (Elsevier, 2012)