EDV (Estruturas de Dados com Vetores) um protótipo de aplicativo Educacional

Luis Arturo Pérez Lozada[1]

[1]Centro de Matemática, Computação e Cognição – Universidade Federal do ABC
CEP 09210-580 – Santo André – SP – Brasil

luis.arturo@ufabc.edu.br

Abstract. *This paper describe the EDV an educational software prototype that may be used for the learning of the Data Structures and its algorithms. The EDV simulates/shows stacks, queues and ordered sequencial lists using an array of non negative integers (between 0 and 99) of length 10. The EDV inplements the insertion and remotion operations in such structures showing, eventually, cases of overflow or underflow.*

Resumo. *Este artigo tem por objetivo apresentar um protótipo do aplicativo educacional intitulado EDV siglas de "Estruturas de Dados com Vetores". Trata-se de um Objeto de Aprendizagem destinado aos discentes da disciplina de Estruturas de Dados e seus Algoritmos. O EDV simula/exibe as estruturas: Pilha, Fila e Les (Lista Estática Seqüencial Ordenada), utilizando um vetor de memória de até 10 posições para o armazenamento de inteiros não negativos de até dois dígitos (entre 0 e 99 inclusive). O EDV implementa as operações de inserção-remoção de elementos em tais estruturas alertando eventuais ocorrências dos estouros positivos e negativos (overflow e underflow).*

1. Cenário de Uso

As Estruturas de Dados são recursos computacionais que permitem armazenar, consultar e recuperar informações de maneira computacionalmente eficiente tanto em tempo quanto em espaço de armazenamento; para maiores detalhes sugerimos consultar Cormen et. al. (2002). A disciplina de Estruturas de Dados e seus Algoritmos é de vital importância em cursos da área de Computação. O *EDV* foi concebido como um Objeto de Aprendizagem com o propósito de oferecer aos discentes, dos cursos superiores da área de computação, de um software livre de suporte ao aprendizado das principais Estruturas de Dados implementadas com Vetores.

2. Desenvolvimento

A metodologia de desenvolvimento de software adotada neste protótipo *EDV* foi a metodologia tradicional em cascata, utilizando o paradigma de Orientação a Objetos. A figura 1 ilustra um diagrama onde as principais classes do protótipo foram concebidas segundo o padrão de Arquitetura de Software MVC (*Model*, *View*, *Controler*). Já a

DOI: 10.5753/cbie.wcbie.2015.256

figura 2 ilustra o Diagrama de Classes do protótipo. Pode-se apreciar nesta figura as relações de herança entre as classes, assim como os atributos e métodos de cada uma das classes constantes na figura 1. Para maiores informações do padrão MVC e do Diagrama de Classes recomenda-se a leitura de Gamma et. al. (2000) e Fowler (2000), respectivamente.

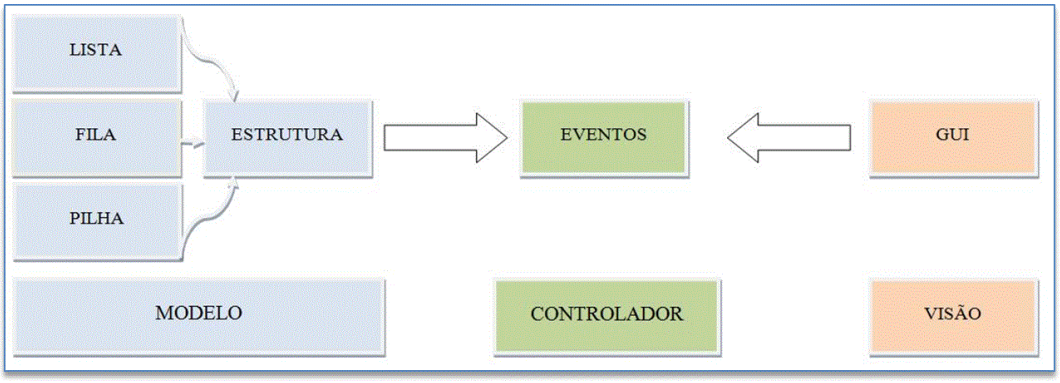

Figura 1: Principais classes do protótipo EDV segundo o padrão MVC.

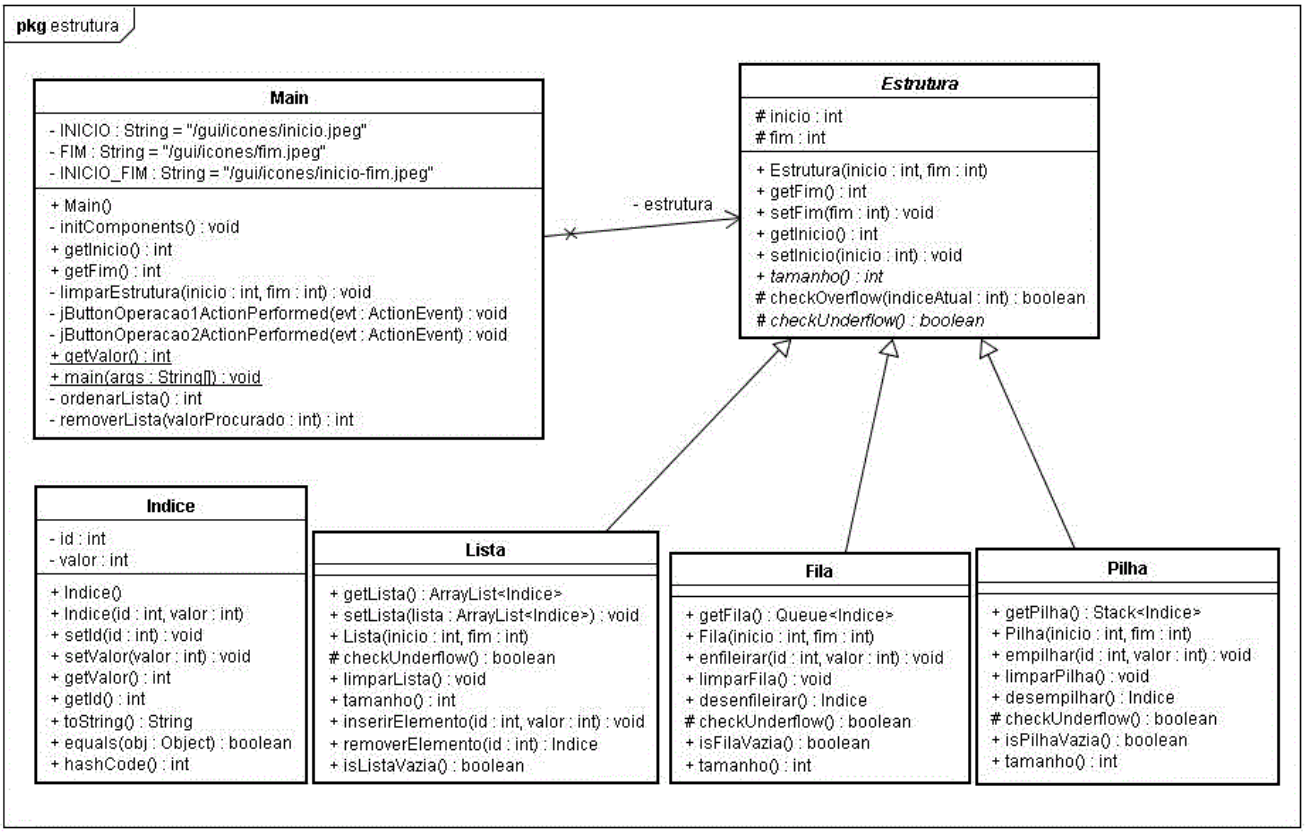

Figura 2: Diagrama de Classes do *EDV*.

O *EDV* foi implementado na linguagem de programação Java utilizando as classes da biblioteca ***javax.swing.****, conjuntamente com as interfaces para o tratamento de eventos da biblioteca ***java.awt.**** . A implementação das classes: Lista, Pilha e Fila,

no *EDV,* herdam propriedades e atributos das classes ***ArrayList***, ***Stack*** *e* ***Queue***, da biblioteca ***java.util.****,* respectivamente. Uma boa referencia bibliográfica para a linguagem Java pode ser obtida em Deitel e Deitel (2010). Não foi realizada a validação formal dos requisitos funcionais de software no protótipo *EDV*.

3. Apresentação do software

O *EDV* executa em modo *stand-alone*; sua interface gráfica (vide figura 3) é bastante simples com uma barra de menus simplificada (**Arquivo** e **Créditos**), uma área principal de simulação/visualização da estrutura e por último, na parte inferior da interface gráfica, uma área de exibição de mensagens e alertas do protótipo.

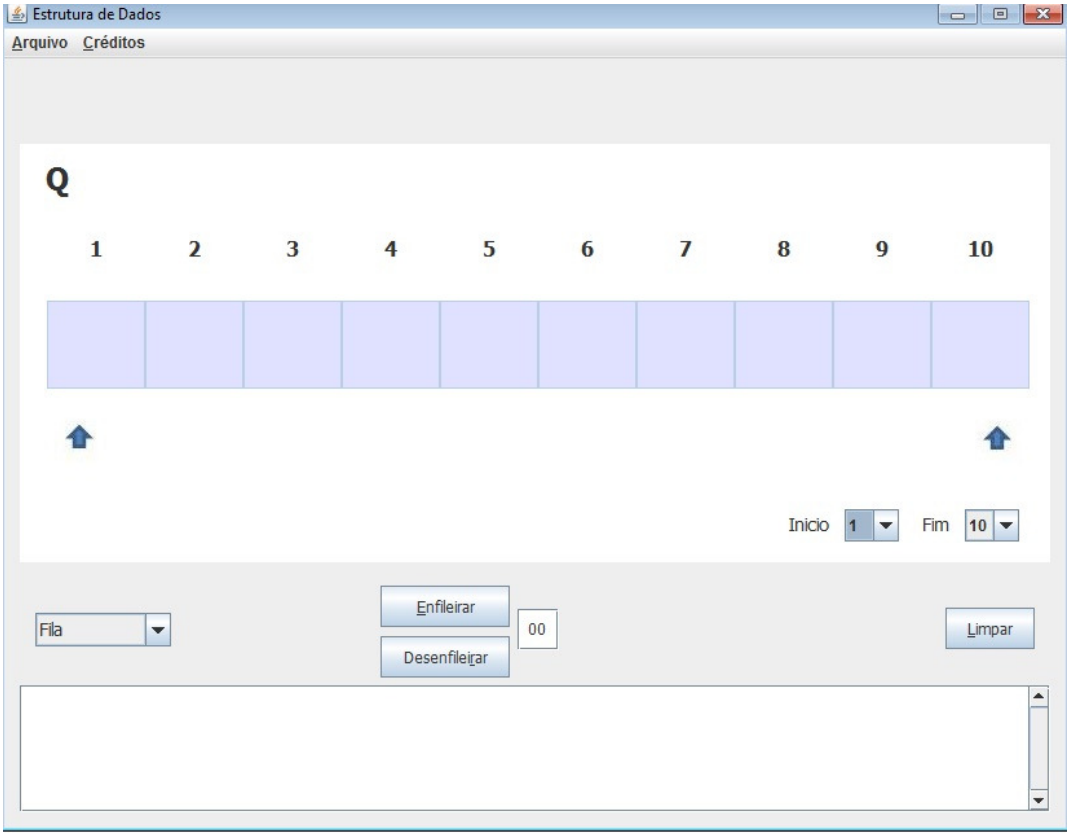

Figura 3: Interface Gráfica do *EDV*.

O *EDV* simula/exibe as estruturas: Pilha, Fila e Les (Lista Estática Seqüencial Ordenada). Vide na figura 4 o componente caixa de lista com opções para cada uma destas estruturas de dados. O *EDV* utiliza um vetor de memória de até 10 posições para o armazenamento de inteiros não negativos de até dois dígitos (entre 0 e 99 inclusive). É possível definir um vetor de tamanho menor a 10 posições, para isso escolha nas caixas de lista correspondentes o início e fim do vetor, por *default* o tamanho é de 10 posições (vide figura 4).

O *EDV* implementa/exibe as operações de inserção-remoção de elementos em Pilhas, Filas e Listas; inclusive alertando eventuais ocorrências dos estouros positivos e negativos (*overflow* e *underflow*) em tais estruturas (vide figuras 5 e 6).

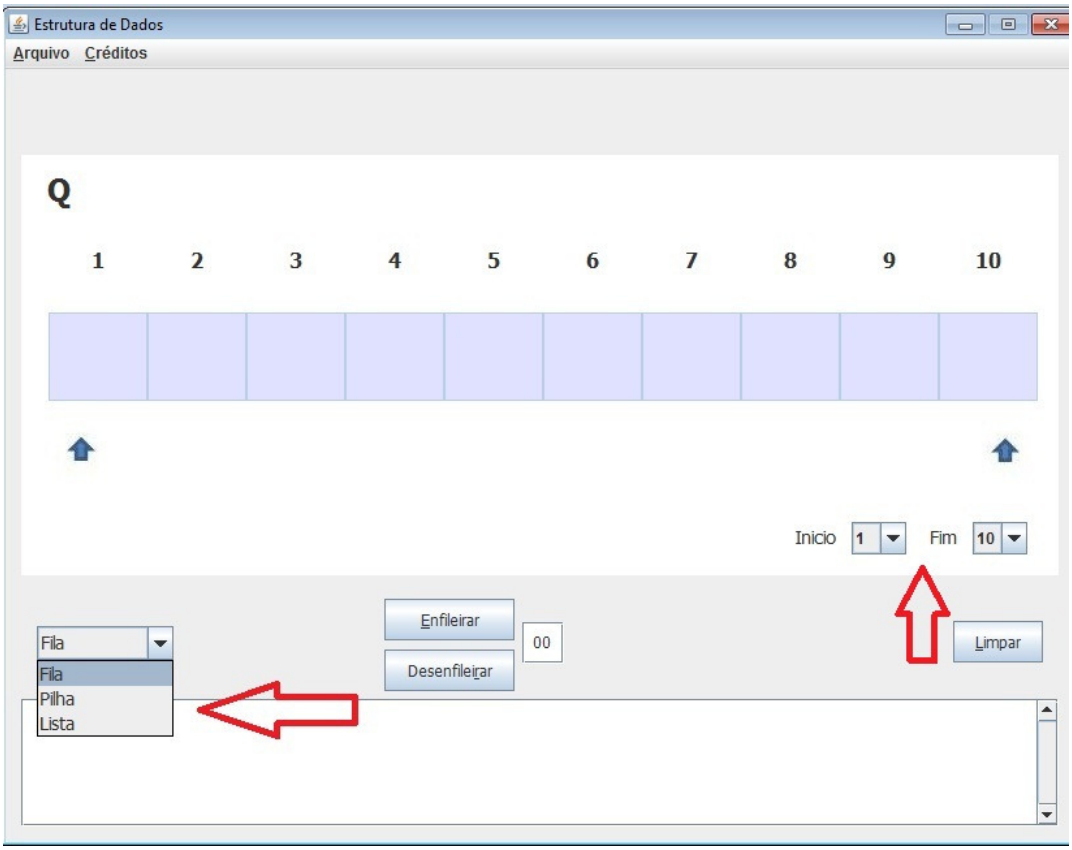

Figura 4: Componentes caixa de lista para escolha da estrutura de dados e para a definição do tamanho do vetor.

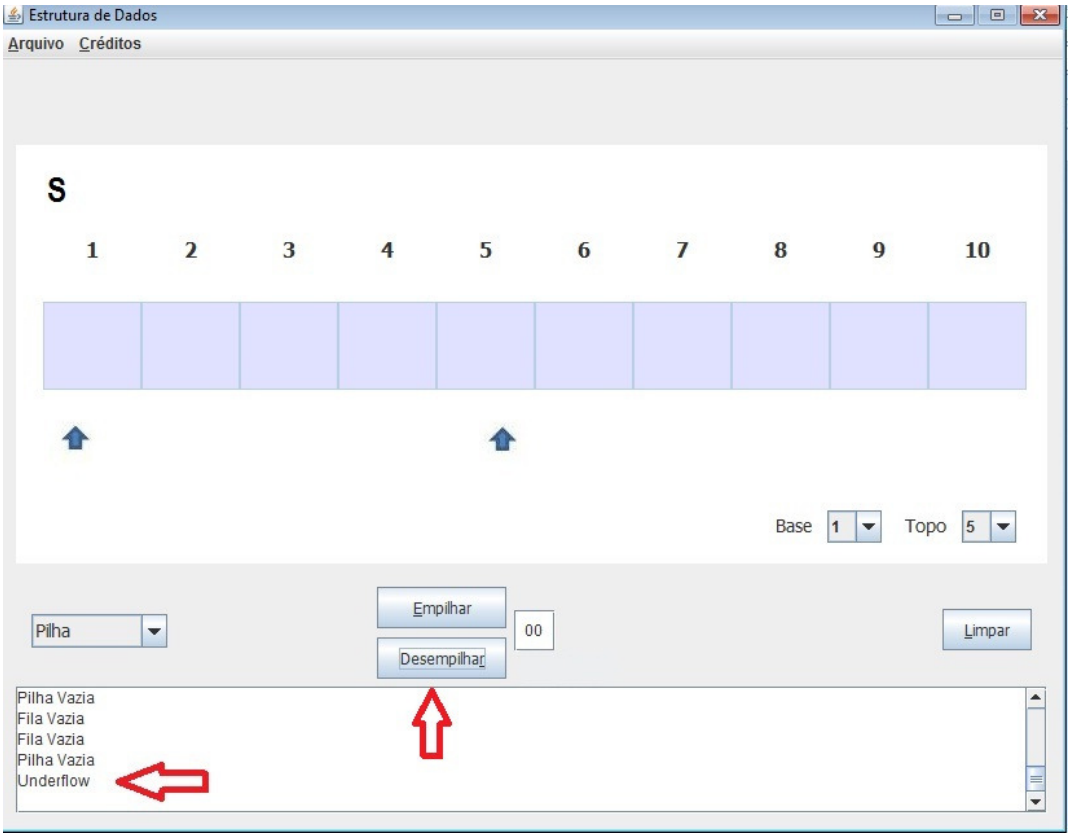

Figura 5: Estouro negativo (underflow) ao desempilhar um elemento numa pilha vazia.

Em particular, a figura 5 ilustra a simulação de uma Pilha implementada num vetor de tamanho 5. Observe que a estrutura inicialmente está "vazia" ou *empty* e após a remoção de um elemento (inexistente) ou "desempilhamento" acontece o "estouro negativo" ou *underflow* na Pilha. A ocorrência deste evento é alertada na área de exibição de mensagens na interface do protótipo.

Analogamente, a figura 6 ilustra a simulação de uma Fila implementada num vetor, também, de tamanho 5. Observe a inserção ou "enfileiramento" consecutivo de 5 inteiros fazendo com que a Fila fique "cheia" ou *full*. Os elementos na fila são exibidos no vetor com fundo azul. Observe que o enfileiramento adicional do inteiro "16" na Fila "cheia" causará o evento "estouro positivo" ou *overflow* na Fila. Esta ocorrência é alertada textualmente na área de exibição de mensagens do software e visualmente no vetor sob um fundo vermelho, indicando desta maneira que o inteiro "16" esta armazenado no vetor, podendo ser indexado através do índice 6, isto é, Q[6]=16, porém este intero não está armazenado na Fila.

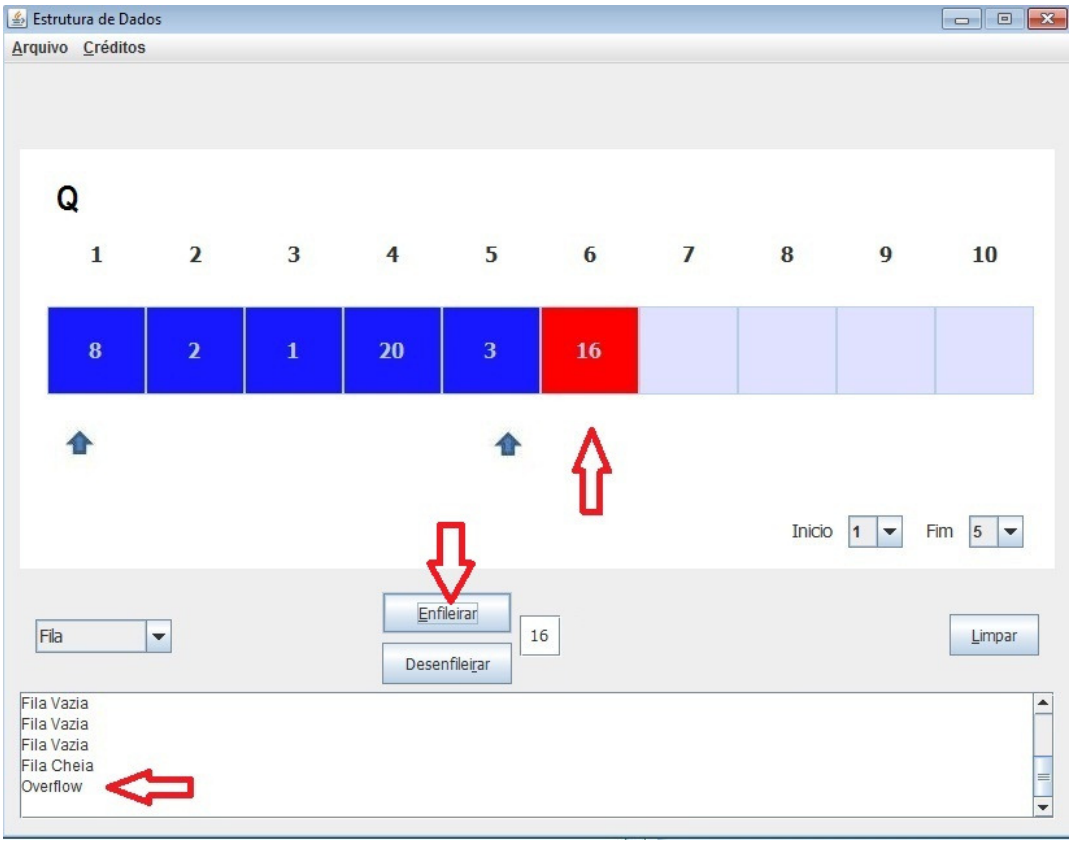

Figura 6: Estouro positivo (overflow) ao enfileirar um elemento numa Fila cheia.

A versão atual do protótipo *EDV* possui um erro de implementação ainda não removido; especificamente relacionado ao funcionamento da estrutura Fila. As figuras 7, 8 e 9 ilustram este *bug*. Considere uma Fila vazia e o enfileiramento, por exemplo, dos inteiros "45" e "55" (vide figura 7). Na seqüência remova um elemento da Fila (vide figura 8). Por último enfileire, por exemplo, o inteiro "65". Observe que o enfileiramento não acontece no final da Fila, mas no início (vide figura 9), constituindo um erro conceitual na implementação da estrutura Fila.

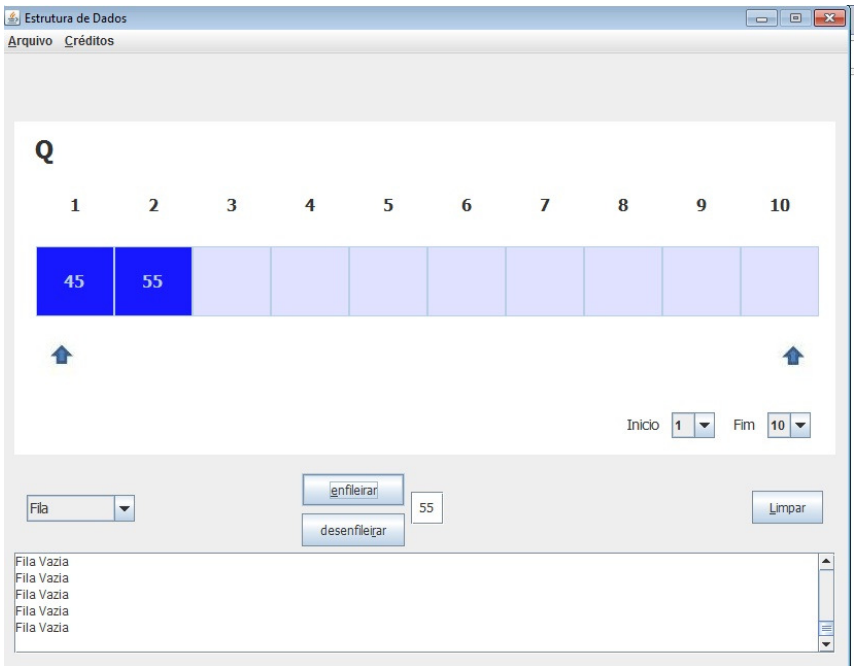

Figura 7: Enfileirando 45 e 55 numa Fila vazia.

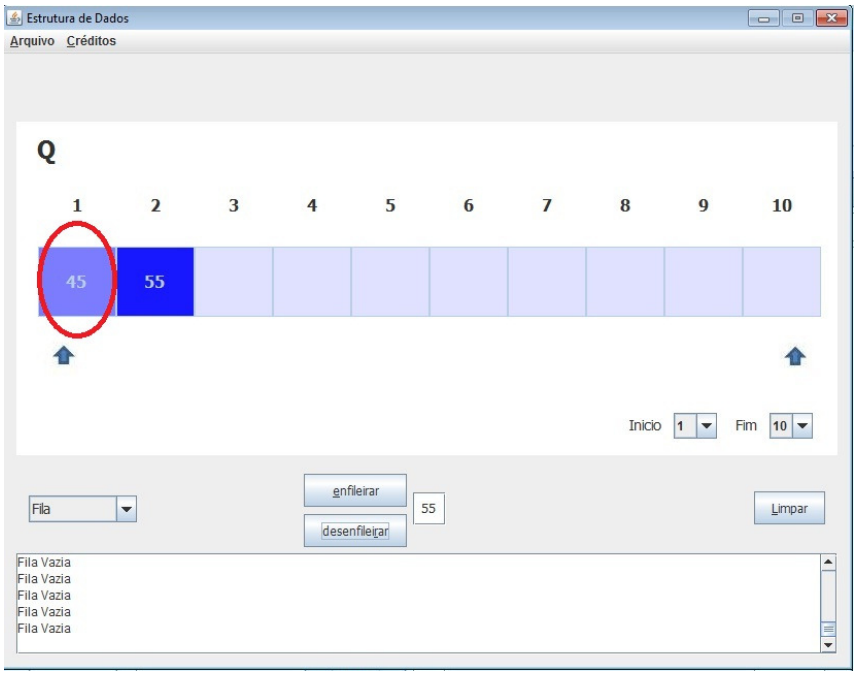

Figura 8: Desenfileirando, na sequência, um elemento da Fila.

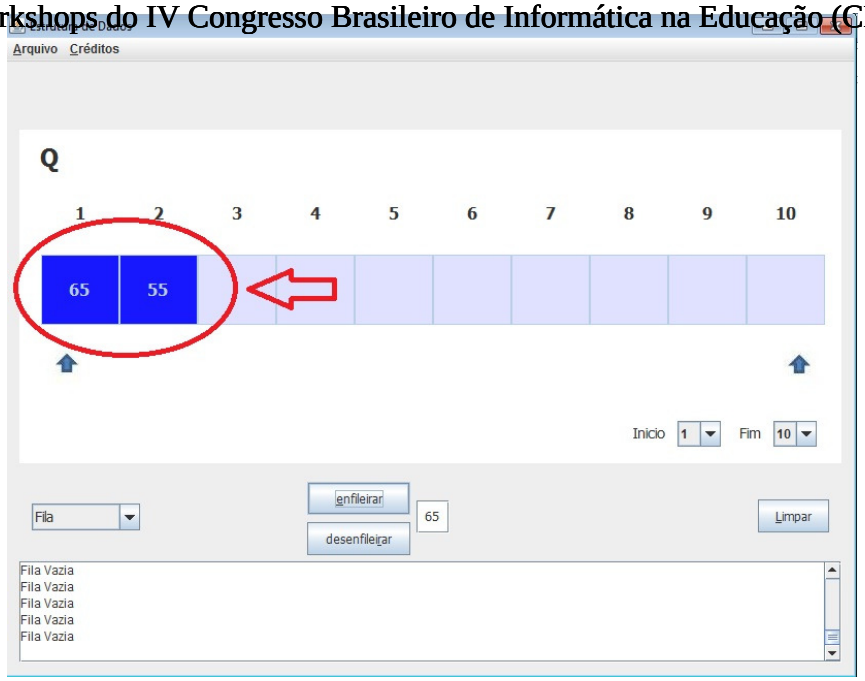

Figura 9: Erro conceitual na implementação de Filas ao inserir um novo elemento após remoções.

4. Considerações finais

Por se tratar de um protótipo estão previstas diversas extensões a fim de tornar-lo um Aplicativo Educacional mais funcional e completo. Entre as principais extensões mencionamos:

- Aprimoramento da Interface Gráfica voltado ao Usúario (GUI) ;
- Implementação da remoção de um determinado item da Les (Lista Estática Sequencial Ordenada); atualmente a remoção sempre acontece no final da Lista;
- Implementação do aplicativo *EDV* para dispositivos movéis (versão *android*);
- Validação formal dos requisitos funcionais de software.

Com a extensões acima listadas acreditamos que o protótipo *EDV* possa ser utilizado como Objeto de Aprendizagem na disciplina de Estrutura de Dados e seus Algoritmos. Inicialmente, não está previsto tornar este protótipo um aplicativo comercial, a não ser, na futura versão *android*.

Referencias

Cormen, Thomas H. et. al. (2002), Algoritmos: teoria e prática. Edt. Elsevier, 2ª Edição, 5ª reimpressão, Rio de Janeiro.

Deitel, H., Deitel, P. (2010), Java: como programar. Prentice-Hall, 8ª edição, São Paulo.

Fowler, M. (2000), UML Essencial: Um breve guia para a linguagem padrão de modelagem de objetos, Bookman, Porto Alegre.

Gamma, Erich et. al. (2000), Padrões de Projeto: Soluções reutilizáveis de software Orientado a Objetos, Bookman, Porto Alegre.