Fila – Navegador e Sistema Hospitalar

1- Introdução

O trabalho desenvolvido consiste em dois programas que abordam diferentes tipos de problemas: O primeiro programa simula a navegação em um navegador básico, permitindo ao usuário acessar, voltar e avançar entre páginas web utilizando uma estrutura de pilha. O segundo programa é uma aplicação gráfica que simula o processo de geração e chamada de senhas em um hospital. Ambos os programas fazem uso de estruturas de dados como pilhas e filas para resolver problemas práticos.

2- Implementação

Navegador: O sistema simula a navegação em páginas web usando duas pilhas: uma para armazenar o histórico de páginas visitadas (voltar) e outra para gerenciar as páginas que podem ser acessadas ao avançar. O programa permite acessar novas páginas, voltar para páginas anteriores e avançar para páginas seguintes.

Principais funções utilizadas:

- **acessarPag(String novaPagina)**: Registra a página atual, adicionando-a à pilha de pilhaVoltar e limpa a pilha de avancarPilha.
- **voltar()**: Move a página atual para a pilha de avancarPilha e recupera a última página da pilha de pilhaVoltar como a nova página atual.
- **avancar()**: Move a página atual para a pilha de pilhaVoltar e recupera a última página da pilha de avancarPilha como a nova página atual.
- **exibirEstado()**: Exibe o estado atual do navegador, mostrando a página atual e se há páginas na pilha de pilhaVoltar ou avancarPilha para voltar ou avançar.

Código Fonte:

```java
import java.util.Stack;

public class Navegador {
    private Stack<String> pilhaVoltar; // Pilha para armazenar páginas anteriores
    private Stack<String> avancarPilha; // Pilha para armazenar páginas futuras
    private String paginaAtual; // Página atualmente aberta

    public Navegador() {
        this.pilhaVoltar = new Stack<>();
        this.avancarPilha = new Stack<>();
        this.paginaAtual = null;
    }

    // Método para navegar para uma nova página
    public void acessarPag(String novaPagina) {
        if (paginaAtual != null) {
            pilhaVoltar.push(paginaAtual); // Adiciona a página atual à pilha de voltar
        }
        paginaAtual = novaPagina; // Define a nova página como atual
        avancarPilha.clear(); // Limpa a pilha de avançar ao acessar uma nova página
        exibirEstado();
    }

    // Método para voltar para a página anterior
    public void voltar() {
        if (!pilhaVoltar.isEmpty()) {
            avancarPilha.push(paginaAtual); // Salva a página atual na pilha de avançar
            paginaAtual = pilhaVoltar.pop(); // Recupera a página anterior
        }
        exibirEstado();
    }

    // Método para avançar para a próxima página
    public void avancar() {
        if (!avancarPilha.isEmpty()) {
            pilhaVoltar.push(paginaAtual); // Salva a página atual na pilha de voltar
            paginaAtual = avancarPilha.pop(); // Recupera a próxima página
        }
        exibirEstado();
    }

    // Método para exibir o estado atual do navegador
    private void exibirEstado() {
        System.out.println("Página atual: " + (paginaAtual != null ? paginaAtual : "Nenhuma"));
        System.out.println("Voltar: " + (!pilhaVoltar.isEmpty()));
        System.out.println("Avançar: " + (!avancarPilha.isEmpty()));
    }

    Run | Debug
    public static void main(String[] args) {
        Navegador nav = new Navegador();
        for (String pagina : new String[]{"x.com", "facebook.com", "ulife.com.br"}) { // Lista de sites que serão verificados
            nav.acessarPag(pagina);
        }
        nav.voltar();
        nav.avancar();
    }
}
```

Sistema hospitalar: A aplicação gera senhas sequenciais e as armazena em uma fila. O sistema permite ao usuário gerar novas senhas e chamar a próxima da fila, exibindo o histórico das senhas geradas e chamadas em uma interface gráfica.

Principais funções utilizadas:

- **Hospital()**: Inicializa a interface gráfica do sistema de geração de senhas, configurando a janela, os botões de gerar senha e chamar próxima senha, além de configurar a lista de histórico e o label para exibir a senha atual.
- **btnGerarSenha.addActionListener(ActionListener e)**: Evento acionado ao clicar no botão "Gerar Senha". Gera uma nova senha sequencial, adiciona essa senha à fila de senhas e ao histórico, e incrementa o contador para a próxima senha.

- **btnChamarProximo.addActionListener(ActionListener e)**: Evento acionado ao clicar no botão "Chamar Próximo". Chama a próxima senha da fila, atualiza o label para exibir a senha atual e adiciona essa ação ao histórico. Caso a fila esteja vazia, exibe uma mensagem de aviso.

Código Fonte:

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.LinkedList;
import java.util.Queue;

public class Hospital extends JFrame {
    private int contador = 1; // Contador para gerar senhas sequenciais
    private Queue<Integer> filaSenhas = new LinkedList<>(); // Fila de senhas a serem chamadas
    private DefaultListModel<String> historicoModel = new DefaultListModel<>(); // Modelo para armazenar o histórico
    private JLabel lblSenhaAtual; // Label para exibir a senha atual
    private JList<String> historicoList; // Lista para exibir o histórico de chamadas

    public Hospital() {
        setTitle(title:"Gerador de Senhas - Hospital");
        setSize(width:400, height:300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        JButton btnGerarSenha = new JButton(text:"Gerar Senha");
        JButton btnChamarProximo = new JButton(text:"Chamar Próximo");
        lblSenhaAtual = new JLabel(text:"Senha Atual: Nenhuma");
        historicoList = new JList<>(historicoModel);

        // Evento para gerar uma nova senha e adicioná-la à fila
        btnGerarSenha.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                filaSenhas.add(contador);
                historicoModel.addElement("Senha gerada: " + contador);
                contador++;
            }
        });

        // Evento para chamar a próxima senha da fila, caso exista
        btnChamarProximo.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (!filaSenhas.isEmpty()) {
                    int senhaChamada = filaSenhas.poll(); // Remove a primeira senha da fila
                    lblSenhaAtual.setText("Senha Atual: " + senhaChamada);
                    historicoModel.addElement("Senha chamada: " + senhaChamada);
                } else {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"Nenhuma senha na fila.");
                }
            }
        });

        // Adiciona os componentes à interface gráfica
        add(btnGerarSenha);
        add(btnChamarProximo);
        add(lblSenhaAtual);
        add(new JScrollPane(historicoList));
        setVisible(b:true);
    }

    Run | Debug
    public static void main(String[] args) {
        new Hospital(); // Inicializa a interface gráfica
    }
}
```

3 - Listagem de testes executados

Navegador:

- **Teste de Acesso a Nova Página**
  Foi testado o método acessarPag, responsável por registrar a navegação para uma nova página. Ao navegar para diferentes sites, o sistema corretamente adicionou a página atual à pilha de páginas anteriores (pilhaVoltar), limpou a pilha de páginas futuras (avancarPilha) e atualizou a página atual. O histórico foi mantido conforme esperado. Além disso, foi verificado que o método funciona corretamente ao navegar para várias páginas e garantir que as pilhas sejam manipuladas corretamente

- **Teste de Voltar para a Página Anterior**
  O método voltar foi testado para garantir que, ao pressionar o botão de voltar, o sistema recuperasse corretamente a página anterior da pilha pilhaVoltar. O comportamento foi verificado e, quando a pilha não estava vazia, o sistema corretamente restaurava a página anterior. Se a pilha estava vazia, o sistema não alterava a página atual. O histórico foi atualizado conforme esperado.

- **Teste de Avançar para a Próxima Página**
  O método avancar foi testado para garantir que, ao pressionar o botão de avançar, o sistema recuperasse a próxima página da pilha avancarPilha. Ao avançar para páginas anteriores, a pilha de páginas anteriores foi corretamente atualizada. Quando não havia páginas futuras, a navegação para frente não foi permitida, como esperado.

- **Teste de Navegação Completa**
  Foi testado o fluxo completo de navegação, passando por várias páginas e utilizando os botões de voltar e avançar. O sistema foi capaz de navegar corretamente para novas páginas, voltar para páginas anteriores e avançar para páginas futuras quando possível. O histórico foi atualizado adequadamente em cada ação.

Sistema hospitalar:

- **Teste de Geração de Senha**
  O método de geração de senhas foi testado ao pressionar o botão "Gerar Senha". O sistema gerou senhas sequenciais a partir de 1, adicionando-as corretamente à fila de senhas (filaSenhas). O contador foi incrementado corretamente a cada nova senha gerada. Além disso, o histórico de senhas geradas foi atualizado com a informação da senha gerada.

- **Teste de Chamada de Senha**
  O método btnChamarProximo foi testado para verificar se o sistema removia a senha da fila corretamente e a exibia no rótulo lblSenhaAtual. O histórico também foi atualizado com a senha chamada. Quando a fila estava vazia, uma mensagem de erro foi exibida, como esperado. Esse comportamento foi validado em diferentes cenários, incluindo quando a fila estava vazia.

- **Teste de Fila Vazia**
  Foi testado o comportamento do sistema quando não havia senhas na fila. Ao pressionar o botão "Chamar Próximo" quando a fila estava vazia, o sistema exibiu corretamente uma mensagem de erro, informando ao usuário que não havia senhas para chamar. Isso foi validado para garantir que o sistema lida com a situação adequadamente.

- **Teste de Interface Gráfica**
  A interface gráfica foi testada para garantir que todos os componentes estivessem funcionando corretamente, incluindo os botões "Gerar Senha" e "Chamar Próximo", o rótulo lblSenhaAtual, e a lista JList de histórico. Ao clicar nos botões, o sistema respondeu corretamente e a interface foi atualizada conforme esperado. O histórico foi exibido corretamente na interface, e as senhas geradas e chamadas estavam visíveis na lista.

4- Conclusão

Este trabalho envolveu a implementação de dois sistemas: um navegador com funcionalidades de voltar e avançar entre páginas usando pilhas e um sistema de gerenciamento de senhas para um hospital, utilizando uma fila para controlar a ordem de chamadas.

- No **sistema de navegação**, o uso de **pilhas** foi eficaz para gerenciar o histórico de páginas e permitir a navegação entre elas. A principal dificuldade foi garantir que o sistema funcionasse corretamente ao acessar páginas anteriores ou avançar quando não havia histórico, o que foi resolvido com verificações adequadas.

- No **sistema de senhas do hospital**, a interface gráfica em **Swing** foi utilizada para gerar e chamar senhas, e a **fila** garantiu a ordem de atendimento. A maior dificuldade foi garantir a atualização dinâmica da interface e o correto gerenciamento dos casos de fila vazia.

5- Bibliografia

- Documentação do Java: https://docs.oracle.com/javase/
- Java Stack (Estrutura de Dados) - GeeksforGeeks: https://www.geeksforgeeks.org/stack-data-structure/
- Java Swing Tutorial – Oracle: https://docs.oracle.com/javase/tutorial/uiswing/
- Swing Tutorial – JavaPoint: https://www.javatpoint.com/java-swing
- CFB Cursos. **Como usar Pilha em Java** (2021). Canal CFB Cursos: https://www.youtube.com/watch?v=nRKZ4SdYfXo