ATIVIDADE FINAL | 3SI

Em Equipe de **três** componentes. Inscreva-se na equipe pelo AVA. **Eventuais** exceções serão tratadas à parte. A equipe deverá **identificar** seus componentes (autores) informando RA e NOME COMPLETO no início dos arquivos `.java` na forma de **comentário** em bloco.

Exemplo,

```
/*
 * Disciplina Programação Orientada a Objetos
 * Autores    Beltrano Camargo   RA 987654
 *            Fulano da Silva    RA 123456
 *            Sicrano de Souza   RA 839275
 * Atividade  Final 3SI
 * Data       11/06/2024
 */
```

Definição

A partir da superclasse **abstrata** *`EstrategiaFIFO`*, da classe `Senha` e do enum `TipoLista`, disponibilizadas em arquivos `.java`, estendendo a classe *`EstrategiaFIFO`*, implemente a subclasse **`Fila`**, em Java, com seus **atributos** e **métodos**, e seu **construtor**, de acordo com a modelagem e especificações a seguir. Não esqueça de **testá-la** e eliminar eventuais erros, antes de finalizar e entregar.

Sistema de Controle de Atendimento

A subclasse `Fila` será utilizada como base do **Sistema de Controle de Atendimento** do Consultório Dr. No Problem, CRM 01.892. A interação humano-computador poderá ser realizada via `console` **ou** `GUI`, à escolha da equipe, em função da sua experiência e conhecimento de Java. **Opcionalmente**, o sistema poderá armazenar os dados necessários ao seu funcionamento no SGBD-R MySQL, v.8. O nome da classe que implementará o Sistema de Controle de Atendimento deve ser **`ControleDeFila`**.

Avaliação do Funcionamento

A equipe deverá **apresentar** ao professor, o sistema **em funcionamento**, como parte da avaliação desta atividade, a partir de **28**/05 até **12**/06/2024.

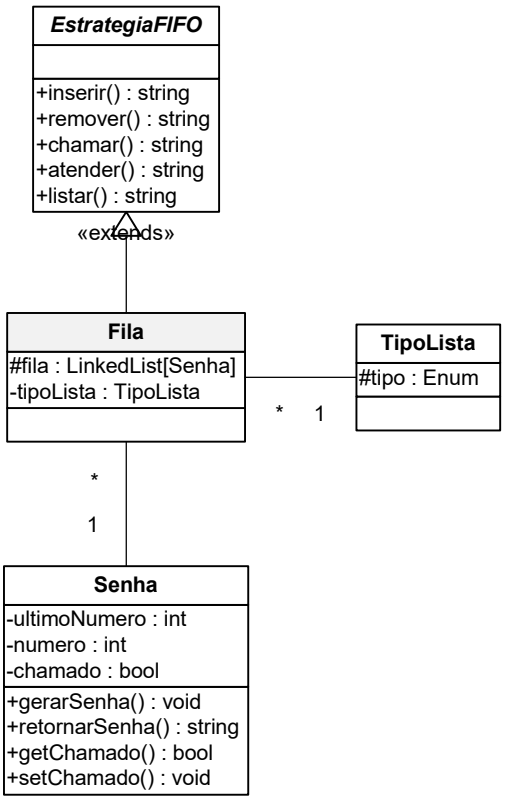
Entrega Final

A equipe deverá postar, como resposta a esta atividade, os arquivos correspondentes a subclasse `Fila` e a classe do sistema **`ControleDeFila`**, em arquivo `.java` desenvolvidos, implementados e testados conforme solicitado.

Especificação

Classes pré-definidas *`EstratégiaFIFO`* | `TipoLista` | `Senha`

UML



Lembrete utilizar **todos** os conceitos sobre **POO** aprendidos e **relevantes** à resolução desta atividade.

Enum `TipoLista`

Não modifique este enum.

```java
public enum TipoLista {
    //Define o enum TipoLista
    2 usages
    IDOSO("I60"),
    2 usages
    IDOSO80("I80"),
    2 usages
    NORMAL("NML"),
    2 usages
    PREFERENCIAL("PFL"),
    2 usages
    URGENTE("URG"),
    2 usages
    VIP("VIP");

    7 usages
    protected String tipo;

    12 usages
    TipoLista(String tipo){
        this.tipo = tipo.toUpperCase();
    }
}
```

Superclasse Abstrata *`EstrategiaFIFO`*

Não modifique esta classe.

```java
public abstract class EstrategiaFIFO {
    //Define a classe abstrata 'EstrategiaFIFO' a partir do método FIFO

    1 usage  1 implementation
    public abstract String inserir();

    1 usage  1 implementation
    public abstract void remover();

    1 usage  1 implementation
    public abstract String chamar();

    1 usage  1 implementation
    public abstract String atender();

    6 usages  1 implementation
    public abstract String listar();
}
```

Classe Senha

Não modifique esta classe.

```java
public class Senha {
    //Define a classe 'Senha'
    2 usages
    private static int ultimoNumero = 0;          //Registra o último número de senha para
                                                  // objetos da classe - atributo de classe
    3 usages
    private int numero = 0;                       //Registra o número de uma senha específica
    3 usages
    private boolean chamado = false;              //Registra o chamado da senha
    2 usages
    public void gerarSenha() {                    //Gera um número para a senha
        ultimoNumero++;
        this.numero = ultimoNumero;
    }
    8 usages
    public String retornarSenha() {               //Retorna o número da senha formatado
        if(this.chamado)
            return String.format("%04d", this.numero) + "*";
        return String.format("%04d", this.numero);
    }
    6 usages
    public boolean getChamado(){
        return this.chamado;
    }
    2 usages
    public void setChamado(){
        this.chamado = true;
    }
}
```

Classes a serem implementadas e entregues pela Equipe

Subclasse `Fila`

Atributos

`fila`	`LinkedList[Senha]` para controlar as **senhas** geradas
`tipoLista`	enum `TipoLista` que identifica o **tipo** da fila criada

Métodos

Criar os métodos necessários para a operacionalização do **controle de uma fila** utilizando o método **FIFO**.

Classe `ControleDeFila`

Atributos

Definir os **atributos** necessários ao funcionamento correto do sistema.

Métodos

Implementar os **métodos** necessários ao funcionamento do sistema.

O **design** da interface humano-computador (`console` ou `GUI`) fica ao gosto do trio.

Requisitos do Sistema

Requisitos Funcionais

Origem LEGAL | NEGÓCIO | AMBOS

	REQUISITO	SOLICITANTE	ORIGEM
1	Realizar o controle de atendimento pelo método **FIFO**	DR. NO	NEGÓCIO
2	Cumprir a Lei Federal nº **10.048**, de 8 de novembro de 2000 que 'Dá prioridade de atendimento às pessoas que especifica, e dá outras providências.'	DR. NO	LEGAL
3	Cumprir a Lei Federal nº **14.626**, de 19 de julho de 2023 que prevê o '[...] atendimento prioritário em diversos estabelecimentos a pessoas com transtorno do espectro autista ou com mobilidade reduzida e a doadores de sangue [...]'	DR. NO	LEGAL
4	O **nome do consultório** e o **CRM** do médico responsável devem aparecer em todas as telas da interface de usuário	DR. NO	NEGÓCIO
5	O tipo da fila e número da senha devem ser exibidos em cores diferentes para cada tipo de lista. Exemplo, URG-0005	DR. NO	NEGÓCIO

Requisitos Técnicos

Origem LEGAL | NEGÓCIO | AMBOS

	REQUISITO	SOLICITANTE	ORIGEM
1	Utilizar a Linguagem Orientada a Objetos **Java**	TIC	NEGÓCIO
2	Utilizar, se houver tempo, o SGBD-R **MySQL**, v.8.0	TIC	NEGÓCIO
3	Implementar a interface com usuário via `console` **ou** `GUI`	TIC	NEGÓCIO
4	Respeitar os conceitos aprendidos em **IHC**	TIC	NEGÓCIO
5	Validar, sempre que necessário, as entradas de dados	TIC	NEGÓCIO