Manual Técnico para el Sistema de Gestión de Turnos

Ariana Marcela Andrade Bello

Emanuel Esteban Restrepo Patarroyo

Universidad Unicomfacauca

Estructura de Datos

Jimmy Roman Valdes Santacruz

27 de mayo de 2024

Manual Técnico para el Sistema de Gestión de Turnos

Introducción

Este manual técnico proporciona una descripción detallada del código del sistema de gestión de turnos, incluyendo la estructura del proyecto, las clases utilizadas, y el flujo de ejecución. Este manual está dirigido a desarrolladores que deseen entender, mantener y extender el código.

Estructura del Proyecto

El proyecto se organiza en tres archivos Java principales:

1. `Turnero.java`
2. `Metodos.java`
3. `Persona.java`

Detalles de las Clases

Clase `Turnero`

```java
package com.mycompany.turnero;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Turnero {

    public static void main(String[] args) {
        Scanner leer = new Scanner(System.in);
        Metodos metodos = new Metodos();
        int turno = 0;

        int menu = 0;

        do {
            try {
                System.out.println("-----------------MENU AGREGAR TURNO----------------");
                System.out.println("1. Agregar nuevo turno");
                System.out.println("2. Mostrar el turno que sera atendido");
                System.out.println("3. Atender Paciente");
                System.out.println("4. Salir");
                System.out.println("-------------------------------------------------\n");

                System.out.println("Ingrese la opcion a seleccionar");
                menu = leer.nextInt();

                switch (menu) {
                    case 1:
                        metodos.crear();
                        break;
                    case 2:
                        turno += 1;
                        metodos.mostrarTurnos(turno);
                        break;
                    case 3:
                        metodos.atenderPaciente();
                        break;
                    case 4:
                        System.out.println("Gracias por utilizar nuestro programa");
                        break;
                    default:
                        System.out.println("Esta opcion no es valida");
                }
            } catch (InputMismatchException e) {
                System.out.println("Esta opción no es válida. Por favor, ingrese un número.");
                leer.next();
            }
        } while (menu != 4);
    }
}
```

Descripción:

- Función Principal (`main`): Este es el punto de entrada de la aplicación. Aquí se configura el menú interactivo y se manejan las opciones seleccionadas por el usuario.

- **Scanner `leer`:** Utilizado para la entrada del usuario.

- Metodos `metodos`: Instancia de la clase `Metodos` que contiene la lógica para manejar turnos.

- Estructura de Control: Un `do-while` loop para mantener el menú interactivo hasta que el usuario decida salir (`menu != 4`).

- Manejo de Excepciones: Captura `InputMismatchException` para manejar entradas no válidas.

Clase `Metodos`

```java
package com.mycompany.turnero;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Metodos {

    private Queue<Persona> listaNoPrioridad = new LinkedList<>();
    private Queue<Persona> listaPrioridad = new LinkedList<>();

    public void crear() {
        Scanner leer = new Scanner(System.in);

        System.out.println("Ingrese el nombre completo");
        String nombreCompleto = leer.nextLine();

        System.out.println("Ingrese el numero de cedula");
        int cedula = leer.nextInt();

        System.out.println("¿Es una prioridad? ingrese true para SI y false para NO");
        boolean prioridad = leer.nextBoolean();

        Persona persona = new Persona(nombreCompleto, cedula, prioridad);

        if (prioridad == true) {
            this.listaPrioridad.add(persona);
        } else {
            this.listaNoPrioridad.add(persona);
        }
    }

    public void mostrarTurnos(int turno) {
        if (!this.listaPrioridad.isEmpty()) {
            System.out.println("Turno n°" + turno + " Asignado a una persona PRIORITARIA");
            System.out.println("El paciente " + listaPrioridad.peek() + "\nEn espera de ser atendido");
        } else if (!this.listaNoPrioridad.isEmpty()) {
            System.out.println("Turno n°" + turno + " Asignado a una persona NO PRIORITARIA");
            System.out.println("El paciente " + listaNoPrioridad.peek() + "\nEn espera de ser atendido");
        } else {
            System.out.println("Error no hay paciente por atender");
        }
    }

    public void atenderPaciente() {
        if (!this.listaPrioridad.isEmpty()) {
            System.out.println("El paciente " + this.listaPrioridad.peek().toString());
            this.listaPrioridad.remove(this.listaPrioridad.peek());
            System.out.println("Ha sido atendido correctamente");
        } else if (!this.listaNoPrioridad.isEmpty()) {
            System.out.println("El paciente" + this.listaNoPrioridad.peek().toString());
            this.listaNoPrioridad.remove(this.listaNoPrioridad.peek());
            System.out.println("Ha sido atendido correctamente");
        } else {
            System.out.println("Error no hay paciente por atender");
        }
    }
}
```

Descripción:

- Listas de Prioridad: Dos colas (`Queue`), `listaPrioridad` y `listaNoPrioridad`, para manejar pacientes prioritarios y no prioritarios respectivamente.

- Método `crear()`: Solicita al usuario los detalles del paciente y lo agrega a la cola correspondiente.

- Método `mostrarTurnos(int turno)`: Muestra el siguiente turno a ser atendido.

- Método `atenderPaciente()`: Atiende al siguiente paciente en la cola y lo elimina de la lista.

Clase `Persona`

```java
package com.mycompany.turnero;

public class Persona {

    private String nombreCompleto;
    private int cedula;
    private boolean prioridad;

    public Persona(String nombreCompleto, int cedula, boolean prioridad) {
        this.nombreCompleto = nombreCompleto;
        this.cedula = cedula;
        this.prioridad = prioridad;
    }

    public String getNombreCompleto() {
        return nombreCompleto;
    }

    public void setNombreCompleto(String nombreCompleto) {
        this.nombreCompleto = nombreCompleto;
    }

    public int getCedula() {
        return cedula;
    }

    public void setCedula(int cedula) {
        this.cedula = cedula;
    }

    public boolean isPrioridad() {
        return prioridad;
    }

    public void setPrioridad(boolean prioridad) {
        this.prioridad = prioridad;
    }

    @Override
    public String toString() {
        return "\nNombre Completo = " + nombreCompleto + " \nCedula = " + cedula + " \nPrioridad = " + prioridad;
    }
}
```

Descripción:

- Atributos: `nombreCompleto`, `cedula`, y `prioridad`.

- Constructor: Inicializa los atributos.

- Métodos Getters y Setters: Permiten acceder y modificar los atributos.

- Método `toString()`: Representa la información del paciente en formato de cadena.

Flujo de Ejecución

1. El usuario ejecuta la clase `Turnero`.

2. El menú se muestra repetidamente hasta que el usuario selecciona salir (`menu != 4`).

3. Dependiendo de la selección del usuario, se invocan los métodos correspondientes en la clase `Metodos`.

4. Los métodos de la clase `Metodos` manejan la lógica para crear, mostrar y atender pacientes.

Link GitHub

https://github.com/EstebanR05/final-proyect-structure-of-data

Conclusión

Este manual técnico proporciona una guía completa sobre el funcionamiento interno del sistema de gestión de turnos, facilitando su mantenimiento y extensión.