**DOCUMENTACIÓN PROYECTO FINAL ESTRUCTURA DE DATOS**

**APLICACIÓN DE GESTIÓN DE CITAS MEDICAS**

El código proporcionado es una parte de un proyecto en Java que utiliza el entorno de desarrollo NetBeans para crear una interfaz gráfica de usuario (GUI) con Java Swing. Aquí hay una descripción de cada parte del código:

1. Comentarios iniciales:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

Estos comentarios son comentarios de estilo de NetBeans. No afectan el comportamiento del código y son simplemente anotaciones para el desarrollador. Pueden ser modificados según las necesidades del proyecto.

2. Paquete y comentarios:

package proyect.Modelo:

/\*\*

\*

\* @author Usuario

\*/

El código está dentro del paquete proyect.Modelo. Además, hay comentarios generados automáticamente por NetBeans indicando la autoría del archivo.

3. Importaciones:

import javax.swing.JOptionPane;

import java.util.ArrayList;

import proyect.Vistas.menuprincipal;

Se importan las clases necesarias para el proyecto. JOptionPane se utiliza para mostrar cuadros de diálogo, ArrayList es una estructura de datos dinámica, y se importa la clase menuprincipal desde el paquete proyect.Vistas.

**CLASE PRINCIPAL:**

4. Clase principal Proyect:

public class Proyect {

Se define la clase principal llamada Proyect.

5. Método principal main:

public static void main(String[] args) {

Aquí comienza el método principal main, que es el punto de entrada del programa.

6. Creación de una instancia de menuprincipal:

menuprincipal principal = new menuprincipal();

Se crea una instancia de la clase menuprincipal. Esta parece ser una clase que representa la interfaz gráfica principal del programa.

7. Visualización de la interfaz gráfica:

principal.setVisible(true);

Se hace visible la interfaz gráfica representada por la instancia de menuprincipal. Esto indica que la aplicación ha comenzado y está mostrando la ventana principal.

**CLASE CITA:**

Se continua con la creación de la clase llamada “cita”, que está diseñada para gestionar información relacionada citas médicas:

8. Declaración del paquete:

package proyect.Modelo;

9. Importaciones:

import java.util.ArrayList;

import java.util.Scanner;

Se importan las clases ArrayList y Scanner desde el paquete java.util. ArrayList se utiliza para almacenar listas dinámicas de elementos, y Scanner se usa para leer la entrada del usuario.

10. Declaración de la clase cita:

public class cita {

Se define la clase cita.

11. Declaración de variables de instancia:

ArrayList fecha;

ArrayList iddoctor;

ArrayList idpaciente;

ArrayList codservi;

ArrayList codcita;

Se declaran cinco variables de instancia, todas del tipo ArrayList. Estas listas dinámicas se utilizan para almacenar fechas, identificadores de médicos, identificadores de pacientes, códigos de servicio y códigos de citas.

12. Constructor:

public cita(ArrayList fecha, ArrayList iddoctor, ArrayList idpaciente, ArrayList codservi, ArrayList codcita) {

this.fecha = fecha;

this.iddoctor = iddoctor;

this.idpaciente = idpaciente;

this.codservi = codservi;

this.codcita = codcita;

}

Se proporciona un constructor que inicializa las variables de instancia con las listas proporcionadas como argumentos.

13. Métodos getter y setter:

Se proporcionan métodos getter y setter para cada variable de instancia, permitiendo acceder y modificar los valores de estas listas dinámicas.

14. Métodos para ingresar información:

public void Ingresarfecha(String fechas) {

fecha.add(fechas);

}

Se proporcionan métodos para agregar información a las listas, como fechas, códigos, identificadores de médicos, pacientes, etc.

15. Métodos para buscar información:

public String Buscarcodigoc(String codigoc) {

// ...

}

Se implementan métodos para buscar información específica en las listas, devolviendo el valor correspondiente si se encuentra o un mensaje indicando que no existe.

16. Métodos para eliminar información:

public void Eliminarcodser(String codser) {

codservi.remove(codservi.indexOf(codser));

}

Se implementan métodos para eliminar información de las listas basándose en un valor específico.

**CLASE MEDICO:**

* Se continua con la creación de la clase llamada “medico“que extiende la clase “persona”.

17. Declaración del paquete:

package proyect.Modelo;

Este código pertenece al paquete proyect.Modelo.

18. Importación de clases:

import proyect.Modelo.persona;

import java.util.ArrayList;

Se importa la clase persona del mismo paquete y la clase ArrayList del paquete java.util.

19. Declaración de la clase “medico”:

public class medico extends persona {

La clase medico extiende la clase persona. Esto significa que hereda los atributos y métodos de la clase persona y puede agregar o sobrescribir funcionalidades específicas para un médico.

20. Constructor:

public medico(ArrayList nombre, ArrayList apellido, ArrayList sexo, ArrayList tpo, ArrayList id, ArrayList telefono, ArrayList direccion, ArrayList eps, ArrayList sueldo, ArrayList especialidad, ArrayList horario, ArrayList edad) {

super(nombre, apellido, sexo, tpo, id, telefono, direccion, eps, sueldo, especialidad, horario, edad);

}

Se proporciona un constructor que llama al constructor de la clase “persona” (la clase base) con las listas proporcionadas como argumentos.

21. Métodos para ingresar información (override de la clase “persona”):

@Override

public void IngresarNombre(String nombres) {

super.nombre.add(nombres);

}

Se implementan métodos para ingresar información específica de médicos, como nombres, apellidos, sexo, etc., utilizando las listas heredadas de la clase persona.

22. Métodos para eliminar información (override de la clase “persona”):

@Override

public void EliminarNombre(String nombres) {

super.nombre.remove(super.nombre.indexOf(nombres));

}

Se implementan métodos para eliminar información específica de médicos de las listas heredadas de la clase “persona”.

23. Métodos para buscar información (override de la clase “persona”):

@Override

public String BuscarNombre(String nombres) {

// ...

}

Se implementan métodos para buscar información específica de médicos en las listas heredadas de la clase persona.

24. Métodos adicionales específicos de médicos:

@Override

public void Ingresardireccion(String direcciones) {

super.direccion.add(direcciones);

}

Se agregan métodos adicionales específicos de médicos, como ingresar dirección, EPS, sueldo, especialidad, horario, etc.

25. Métodos adicionales para eliminar y buscar información específica de médicos:

@Override

public void Eliminardireccion(String direcciones) {

super.direccion.remove(super.direccion.indexOf(direcciones));

}

Se implementan métodos adicionales para eliminar y buscar información específica de médicos en las listas heredadas de la clase persona.

**CLASE PACIENTE**

* Ahora se continua con la creación de la clase llamada “paciente” que extiende la clase “persona”. Aquí hay una descripción de cada parte del código:

26. Declaración del paquete:

package proyect.Modelo;

Este código pertenece al paquete proyect.Modelo.

27. Importación de clases:

import proyect.Modelo.persona;

import java.util.ArrayList;

Se importa la clase persona del mismo paquete y la clase ArrayList del paquete java.util.

28. Declaración de la clase “paciente”:

public class paciente extends persona {

La clase “paciente“ extiende la clase “persona”, heredando sus atributos y métodos y permitiendo la adición o sobrescritura de funcionalidades específicas para pacientes.

29. Constructor:

public paciente(ArrayList nombre, ArrayList apellido, ArrayList sexo, ArrayList tpo, ArrayList id, ArrayList telefono, ArrayList direccion, ArrayList eps, ArrayList sueldo, ArrayList especialidad, ArrayList horario, ArrayList edad) {

super(nombre, apellido, sexo, tpo, id, telefono, direccion, eps, sueldo, especialidad, horario, edad);

}

Se proporciona un constructor que llama al constructor de la clase persona (la clase base) con las listas proporcionadas como argumentos.

30. Métodos para ingresar información (override de la clase “persona”):

@Override

public void IngresarNombre(String nombres) {

super.nombre.add(nombres);

}

Se implementan métodos para ingresar información específica de pacientes, como nombres, apellidos, sexo, etc., utilizando las listas heredadas de la clase “persona”.

31. Métodos para eliminar información (override de la clase “persona”):

@Override

public void EliminarNombre(String nombres) {

super.nombre.remove(super.nombre.indexOf(nombres));

}

Se implementan métodos para eliminar información específica de pacientes de las listas heredadas de la clase “persona”.

32. Métodos para buscar información (override de la clase “persona”):

@Override

public String BuscarNombre(String nombres) {

// ...

}

Se implementan métodos para buscar información específica de pacientes en las listas heredadas de la clase “persona”.

33. Métodos adicionales específicos de pacientes:

@Override

public void Ingresardireccion(String direcciones) {

super.direccion.add(direcciones);

}

Se agregan métodos adicionales específicos de pacientes, como ingresar dirección, EPS, sueldo, especialidad, horario, etc.

34. Métodos adicionales para eliminar y buscar información específica de pacientes:

@Override

public void Eliminardireccion(String direcciones) {

super.direccion.remove(super.direccion.indexOf(direcciones));

}

Se implementan métodos adicionales para eliminar y buscar información específica de pacientes en las listas heredadas de la clase persona.

**CLASE PACIENTE**

Ahora se continua con la creación de la clase llamada “persona.”

35. Declaración del paquete:

package proyect.Modelo;

Este código pertenece al paquete proyect.Modelo

36. Declaración de la clase abstracta “persona”:

public abstract class persona {

La clase “persona” es declarada como abstracta, lo que significa que no se puede instanciar directamente. Se espera que las clases hijas (como “medico“ y “paciente”) proporcionen implementaciones concretas de los métodos abstractos.

37. Declaración de atributos:

ArrayList nombre, apellido, sexo, tpo, id, telefono, direccion, eps, sueldo, especialidad, horario;

ArrayList edad;

Se declaran varias listas dinámicas (ArrayList) para almacenar información sobre una persona, como nombre, apellido, etc.

38. Constructor:

public persona(ArrayList nombre, ArrayList apellido, ArrayList sexo, ArrayList tpo, ArrayList id, ArrayList telefono, ArrayList direccion, ArrayList eps, ArrayList sueldo, ArrayList especialidad, ArrayList horario, ArrayList edad) {

// Inicialización de los atributos con los parámetros proporcionados.

}

Se proporciona un constructor para inicializar los atributos de la clase con las listas proporcionadas como argumentos.

39. Métodos de acceso (getters y setters):

// Métodos getter y setter para cada atributo.

Se proporcionan métodos de acceso para obtener y establecer los valores de los atributos de la clase.

40. Métodos abstractos:

public abstract void IngresarNombre(String nombres);

Se declaran métodos abstractos que deben ser implementados por las clases hijas. Estos métodos representan operaciones como ingresar información, eliminar información y buscar información, y están diseñados para ser específicos de cada tipo de persona.

41. Métodos concretos no abstractos:

Se podrían agregar métodos concretos no abstractos si hay operaciones que son comunes para todas las personas y no necesitan ser implementadas de manera diferente en las clases hijas.

**CLASE PACIENTE**

Ahora se continua con la creación de la clase llamada “servicio” que representa un servicio en el sistema y tiene la siguiente estructura:

42. Declaración del paquete:

package proyect.Modelo;

Este código pertenece al paquete proyect.Modelo.

43. Importación de bibliotecas:

import java.util.ArrayList;

import java.util.Scanner;

Se importan las clases ArrayList y Scanner desde el paquete java.util, que se utilizan en la implementación de la clase.

44. Declaración de la clase “servicio”:

public class servicio {

La clase “servicio“ representa un servicio y no es abstracta, por lo que puede ser instanciada directamente.

45. Atributos de la clase:

ArrayList nombre\_servicio;

ArrayList valor;

ArrayList codigo;

Se declaran tres listas dinámicas (ArrayList) para almacenar información sobre el servicio, como el nombre, el valor y el código del servicio.

46. Constructor:

public servicio(ArrayList nombre\_servicio, ArrayList valor, ArrayList codigo) {

// Inicialización de los atributos con los parámetros proporcionados.

}

Se proporciona un constructor para inicializar los atributos de la clase con las listas proporcionadas como argumentos.

47. Métodos de acceso (getters y setters):

Se proporcionan métodos de acceso para obtener y establecer los valores de los atributos de la clase.

48. Métodos para operaciones específicas:

public void Ingresarcodigo(String codigos){

codigo.add(codigos);

}

public void Ingresarnombre(String nombresc){

nombre\_servicio.add(nombresc);

}

public void Ingresarvalor(String valores){

valor.add(valores);

}

Métodos para ingresar información sobre el servicio, como código, nombre y valor.

49. Métodos para eliminar información:

public void Eliminarcodigo(String codigos) {

codigo.remove(codigo.indexOf(codigos));

}

public void Eliminarvalor(String valores) {

valor.remove(valor.indexOf(valores));

}

public void Eliminarnombre(String nombresc) {

nombre\_servicio.remove(nombre\_servicio.indexOf(nombresc));

}

Métodos para eliminar información del servicio, como código, nombre y valor.

50. Métodos para buscar información:

public String Buscarnombre(String nombresc) {

// ...

}

public String Buscarvalor(String valores) {

// ...

}

public String Buscarcodigo(String codigos) {

// ...

}

Métodos que devuelven la información correspondiente al nombre, valor y código del servicio si existe, o un mensaje indicando que no existe.

**RESUMEN GENERAL DEL CÓDIGO:**

**1. Clase Proyect (Archivo: Proyect.java):**

**Descripción:** La clase Proyect contiene el método main que crea una instancia de la clase menuprincipal y la hace visible.

**Funcionalidad:** Inicia la aplicación y muestra el menú principal.

**2. Clase cita (Archivo: cita.java):**

**Descripción:** La clase cita gestiona la información relacionada con las citas médicas.

**Atributos:** ArrayList para almacenar fechas, IDs de médicos, IDs de pacientes, códigos de servicios y códigos de citas.

**Métodos:** Ingresar información (fecha, código de cita, etc.).

Buscar información.

Eliminar información.

**Funcionalidad:** Gestiona las citas médicas, permitiendo ingreso, búsqueda y eliminación de información.

**3. Clase medico (Archivo: medico.java):**

**Descripción:** La clase medico extiende la clase persona y representa a los médicos en el sistema.

**Atributos:** Hereda los atributos de la clase persona (nombre, apellido, etc.).

**Métodos:** Métodos de ingreso, eliminación y búsqueda específicos para médicos.

**Funcionalidad:** Gestiona la información de los médicos en el sistema.

**4. Clase paciente (Archivo: paciente.java):**

**Descripción:** La clase paciente extiende la clase persona y representa a los pacientes en el sistema.

**Atributos:** Hereda los atributos de la clase persona (nombre, apellido, etc.).

**Métodos:** Métodos de ingreso, eliminación y búsqueda específicos para pacientes.

**Funcionalidad:** Gestiona la información de los pacientes en el sistema.

**5. Clase persona (Archivo: persona.java):**

**Descripción:** La clase persona es una clase abstracta que representa a las personas en el sistema (médicos y pacientes).

**Atributos:** Listas dinámicas para almacenar información personal y laboral.

**Métodos:** Métodos abstractos para ingreso, eliminación y búsqueda de información personal y laboral.

**Funcionalidad:** Proporciona una base común para las clases medico y paciente.

**6. Clase servicio (Archivo: servicio.java):**

**Descripción:** La clase servicio representa los servicios ofrecidos en el sistema.

**Atributos:** ArrayList para almacenar nombres, valores y códigos de servicios.

**Métodos:** Métodos para ingresar, eliminar y buscar información sobre servicios.

**Funcionalidad:** Gestiona la información de los servicios ofrecidos en el sistema.