Practica 1
Programacion Orientada a Objetos
Programacion En Java

Presentado por:
Daniel Fernando Correa Carreño - *dancorreaca@unal.edu.co*
Brahian Steven Serna Restrepo - *bserna@unal.edu.co*

Profesor:
Jaime Alberto Guzman Luna
jaguzman@unal.edu.co

Miércoles 1 de Junio

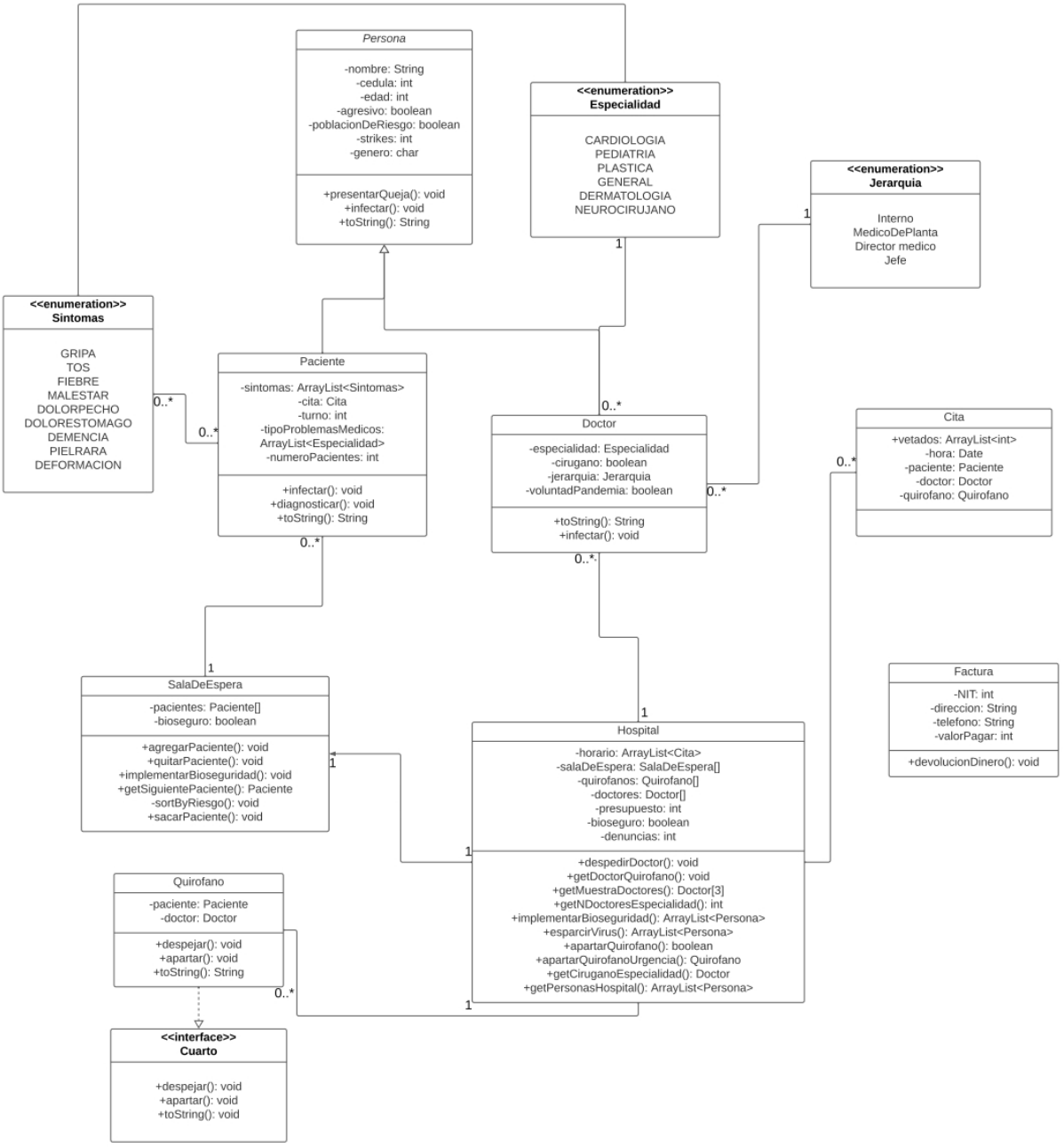
Universidad Nacional de Colombia
Facultad de Minas
2022

CONTENIDO

Descripción general de la solución

CLInical no tiene aplicación en el mundo real y es solo un programa con propósito académico, está contextualizado con el programa para un hospital, no es práctico para la vida real porque las funcionalidades están sobre situaciones imaginarias, por ende no resuelve ningún problema de negocio, administración, etc.

Descripción del diseño estático del sistema en la especificación UML



Clases

- Hospital: Tiene relaciones varias clases entre ellas esta quirofano, porque en apartarQuirofanoUrgencia() se hace uso de tal clase, y la relacion con sala de espera es porque en esparcirVirus() se hace uso de tal clase. Todo hospital tiene sala de espera, quirofanos y doctores

- Quirofano: Hay una relacion con el objeto hospital porque Hospital en sus atributos contiene una lista de Quirofanos. Quirofano implementa la interfaz cuarto
- SalaDeEspera: La sala de espera de un hospital tiene una lista de pacientes y tiene un atributo de tipo booleano para saber si es bioseguro o no. Esta clase tiene multiples metodos para manejar la lista de pacientes que tiene como lo es sortByRiesgo()
- Factura: Utilizado para generar facturas, no esta relacionado con otro objeto, vive de forma independiente ya que cuando se genera una factura solo espera el valor a pagar, que es atributo de instancia. Direccion, NIT y telefono son constantes y son de clase
- Cita: Clase que tiene como atributos fecha, paciente, doctor y quirofano. vetados es una variable estatica
- Doctor: Clase Hija de Persona que dentro de sus atributos tiene referencia a la enumeración Especialidad
- Paciente: Clase Hija de Persona relacionada con la clase SalaDeEspera porque en SalaDeEspera hay varios metodos que utilizan la clase Paciente. numeroPacientes es una variable estatica
- Persona(Clase abstracta): Representa a una persona, el metodo toString es un metodo abstracto, infectar y presentar queja son metodos de instancia

Interfaces

- Cuarto: Implementa los metodos despejar, apartar y el toString

Enumeraciones

- Jerarquia: Es utilizado en el atributo *jerarquia* de la clase Doctor. La enumeracion tiene información sobre el titulo que recibe el doctor
- Sintomas: Es utilizado en el atributo *sintomas* de la clase Paciente. La enumeracion tiene informacion sobre la especialidad del doctor que se debe buscar
- Especialidad: Es utilizado en el atributo *especialidad* de la clase Doctor. La enumeracion tiene informacion sobre el titulo que se le da al doctor, el salario mensual del doctor, y el costo de la cirugia

Descripción de la Implementación de características de programación orientada a objetos en el proyecto

Clase Abstracta y metodos Abstractos:

Fragmento de Codigo del Archivo Persona.java del paquete gestorAplicacion

```
0          /*
1           * ABSTRACT METHODS
2           */
3
4          public abstract String toString();
```

Persona es una clase abstracta que es padre de Paciente y Doctor, en el diseño de nuestro programa es necesario que toda persona tenga un metodo toString definido que sea diferente al de object, por lo que la solucion es declarar el metodo abstracto toString

---

Interfaces:

Fragmento de Codigo del Archivo Cuarto.java del paquete gestorAplicacion

```
1  package gestorAplicacion;
2
3  public interface Cuarto {
4
```

Cuarto es una interface en el programa, esto nos ayuda a imponer un protocolo para los diseños que quieren implementar esta interface como por ejemplo lo hace la clase Quirófano.

Herencia: Persona, Doctor y Paciente son clases que implementan la herencia porque Persona es la clase padre de Doctor y Paciente, es decir, las clases Doctor y Paciente son clases hijas de Persona

Para mostrar un caso puntual del uso de la herencia, voy a exponer el uso que hace el metodo toString de la clase Doctor, el cual utiliza el metodo getNombre(), que utiliza a su vez el atributo privado nombre, que se heredan desde Persona
Para ello considere el siguiente codigo:

Fragmento Codigo del archivo Persona.java del paquete gestorAplicacion

```
5          /* GETTERS & SETTERS */
6          public String getNombre() { return this.nombre; }
```

Fragmento Codigo del archivo Doctor.java del paquete gestorAplicacion

```
1
2 public class Doctor extends Persona {
```

Fragmento Codigo del archivo Doctor.java del paquete gestorAplicacion

```
0       /*
1        * toString: <Jerarquia> <Nombre> de la especialidad <Especialidad>
2        */
3       public String toString() {
4               String output = "";
5
6               if(this.getJerarquia() != null ) {
7                       output += this.getJerarquia().getTitulo();
8               }
9
10              output += "" + this.getNombre() + " de la especialidad " + this.getEspecialidad().getTitulo();
11              return output;
12      }
```

Note que en la linea 10(indexado relativo) del metodo toString de la clase Doctor, se utiliza el this.getNombre() y nunca se declara en el archivo tal metodo, esto es posible porque tal metodo se hereda de la clase Persona, que a su vez, utiliza el atributo this.nombre, el cual nunca se declaro en la clase Doctor, pero de nuevo, esta esta implicitamente porque se hereda de la clase Persona

---

Ligadura Dinámica: Considere el metodo infectar, que se utiliza especialmente en la funcionalidad *Experimento sale mal*
Todas las personas se pueden infectar de tal virus que hace envejecer, por lo que se implementa el metodo en la clase Persona, pero se sobre escribe tanto en Paciente y Doctor

Fragmento de codigo del Archivo Persona.java del paquete gestorAplicacion

```
60      public boolean infectar() {
61          int nuevaEdad = (int) Math.floor(getEdad() * 1.10);
62          setEdad(nuevaEdad);
63          return isPoblacionDeRiesgo();
64      }
```

Fragmento del codigo del Archivo Paciente.java del paquete gestorAplicacion

```
public boolean infectar() {
    boolean retorno = super.infectar();
    addSintomas(Sintomas.DEMENCIA);
    return retorno;
}
```

Fragmento del codigo del Archivo Doctor.java del paquete gestorAplicacion

```
public boolean infectar() {
    boolean retorna = super.infectar();
    if(this.isPoblacionDeRiesgo()) {
        setVoluntadPandemia(false);
    }
    return retorna;
}
```

Cuando se llame a la funcion esparcirVirus() de la clase Hospital, en la linea donde se hace persona.infectar() el puntero es de tipo Persona, y Persona tiene tal metodo pero por ligadura dinamica va a ejecutar ya sea el de Doctor o Paciente

Fragmento del codigo del Archivo Hospital.java del paquete gestorAplicacion

```
public ArrayList<Persona> esparcirVirus() {
    ArrayList<Persona> personasHospital = getPersonasHospital();
    ArrayList<Persona> muertos = new ArrayList<Persona>();
    Random rand = new Random();

    for (Persona persona : personasHospital) {
        if(persona.isPoblacionDeRiesgo() && rand.nextBoolean()) {
            boolean muerto = persona.infectar();
            if(muerto) {
                muertos.add(persona);
                if(persona instanceof Paciente) {
                    this.getSalaDeEspera().sacarPaciente(persona.getCedula());
                } else if(persona instanceof Doctor) {
                    this.despedirDoctor(persona.getCedula(), true);
                }
            }
        }
    }
    return muertos;
}
```

Atributos de Clase y Métodos de clase:

Fragmento de Codigo del Archivo Pacientes.java del paquete gestorAplicacion

```
14      private static int numeroPacientes;
15
```

Para contabilizar el número de pacientes creados fue necesario crear un atributo de clase NumeroPacientes

Fragmento de Codigo del Archivo SubeDoctor.java del paquete Uimain

```
182
183     public static boolean juego(){
184         final int INTENTOS_TOTALES = 7; // Constante con el limite de fallos
185         int intentos = 0;
```

Usamos un método de clase en la clase SubeDoctor para ejecutar estas líneas en las que no interactúa ninguna instancia de la aplicación hecha.

Uso de constantes: En la clase factura, tenemos un NIT, direccion y telefono que vamos a dejar en cada factura y no queremos que cambie, por lo que implementamos el keyword final

Fragmento de Codigo del Archivo Factura.java del paquete gestorAplicacion

```
2 public class Factura {
3
4         /*      CLASS VARIABLES */
5         private static final int NIT = 12;
6         private static final String direccion = "Calle 1 #23";
7         private static final String telefono = "#444 4444 (444)";
```

Encapsulamiento:

- Public: Note por ejemplo el caso para el atributo jerarquia de la clase Doctor

Fragmento de codigo del Archivo Doctor.java del paquete gestorAplicacion

```
8     public Jerarquia jerarquia;
```

- Protected: Note el caso de los atributos nombre y cedula de la clase Persona

Fragmento de codigo del Archivo Persona.java del paquete gestorAplicacion

```
3  public abstract class Persona {
4
5      // INSTANCE ATTRIBUTES
6      protected String nombre;
7      protected int cedula;
```

Note ademas que como Paciente hereda de persona y por ende nombre y cedula son visibles, se puede emplear en el metodo toString de la siguiente forma

```
88     public String toString() {
89         return "Paciente " + nombre + " identificado con documento " + cedula;
90     }
```

- Private: Gran parte de los atributos de las clases son privados y para acceder a ellos se hace por medio de getters & setters

Fragmento de codigo del Archivo Persona.java del paquete gestorAplicacion

```
9      private boolean poblacionDeRiesgo;
10     private boolean agresivo;
11     private int strikes;
12     private char genero;
```

Sobrecarga de metodos y constructores:

- Sobrecarga de constructores: En varios archivos se hace uso de la sobrecarga de constructores, note por ejemplo el caso de Persona

Fragmento de codigo del Archivo Persona.java del paquete gestorAplicacion

```java
    // CONSTRUCTORS
    public Persona(String nombre, int cedula) {
        this(nombre, cedula, false, 0);
    }

    public Persona(String nombre, int cedula, boolean poblacionDeRiesgo) {
        this(nombre, cedula, poblacionDeRiesgo, 0);
    }

    public Persona(String nombre, int cedula, int strikes) {
        this(nombre, cedula, false, strikes);
    }

    public Persona(String nombre, int cedula, boolean poblacionDeRiesgo, int strikes) {
        this.nombre = nombre;
        this.cedula = cedula;
        this.poblacionDeRiesgo = poblacionDeRiesgo;
        this.strikes = strikes;
    }
```

Uso del this(): Considere el siguiente codigo y note que en varios constructores se utiliza el this() que llama al constructor mas general

Fragmento del Codigo del Archivo Paciente.java del paquete gestorAplicacion

```java
    /*      CONSTRUCTORS    */
    public Paciente(String nombre, int cedula) {
            this(nombre, cedula, false, null, null);
    }
    public Paciente(String nombre, int cedula, boolean poblacionDeRiesgo) {
            this(nombre, cedula, poblacionDeRiesgo, null, null);
    }
    public Paciente(String nombre, int cedula, boolean poblacionDeRiesgo,Sintomas sintoma) {
            super(nombre, cedula, poblacionDeRiesgo);
            this.sintomas.add(sintoma);
    }

    public Paciente(String nombre, int cedula, Sintomas sintoma) {
            this(nombre, cedula, false, new Sintomas[]{sintoma}, -1);
    }

    public Paciente(String nombre, int cedula, boolean poblacionDeRiesgo, Sintomas[] sintomas, int turno) {
            super(nombre, cedula, poblacionDeRiesgo, 0);

            for (int i = 0; i < sintomas.length; i++) {
                    this.sintomas.add(sintomas[i]);
            }

            this.turno = turno;
            this.diagnosticar();
    }
```

Implementacion de enumeracion

Fragmento de Codigo del Archivo

```
9         private ArrayList<Sintomas> sintomas = new ArrayList<Sintomas>();
10        private ArrayList<Especialidad> tipoProblemasMedicos = new ArrayList<Especialidad>();
```

Descripción de cada una de las 5 funcionalidades implementadas

Experimento en laboratorio sale mal

En este hospital tan particular de un mundo ficticio, hay un laboratorio lider en investigacion que actualmente se encuentra haciendo experimentos con un nuevo virus que hace envejecer a las personas
La funcionalidad hace que el experimento salga mal y se esparsa el virus por el hospital
Empieza mostrando el estado de la sala de espera
Cuando se esparce el virus, aleatoriamente se escoge a los pacientes y doctores del hospital que son infectados, para los pacientes que se infecten les da demencia, no es el caso de los doctores porque ellos en este mundo ficticio tienen vacunas contra tal virus, si el paciente ademas es poblacion de riesgo muere, y el programa le da un sentido pensame
Se implementa bioseguridad en el hospital, en particular en la sala de espera, se deja silla por medio, antes de eso se ordenan los pacientes, siento los que estan en poblacion de riesgo primero, y en caso de que no halla sillas para todos, se sacan a los ultimos pacientes en la sala de espera y el programa los anuncia en pantalla
Ademas si el paciente hechado es agresivo, se recibe una denuncia por parte de tal paciente

Paciente Urgente Necesita Ayuda

Un paciente ha llegado al hospital y necesita ayuda urgente, el medio de transporte por el que llego es aleatorio

```
Ha llegado alguien en ambulancia
Ha llegado alguien en helicoptero
Ha llegado alguien en carro particular
```

Ejemplos de opciones de medio de transporte

Aleatoriamente el sistema elije un motivo por el que llego al hospital

```
El paciente llego a urgencias porque le dio un paro cardiaco mientras se comia una hamburguesa
El paciente llego a urgencias porque lamio la calle y ahora tiene una tos increible
El paciente llego a urgencias porque se puso un stricker y ahora la piel se ve bastante mal
```

Ejemplos de opciones de motivo por el que llego

Ademas, el motivo por el que llego tambien esta relacionado con la especialidad del doctor que lo va a operar, por ejemplo

Motivo	Especialidad Doctor
Paro cardiaco	Cardiologia
Envenamiento Comida	General

El paciente puede estar identificado o no, y si esta identificado se genera un nombre y un numero de cedula, todo esto es aleatorio

```
El paciente se llama Pedro y esta identificado con cedula de ciudadania 6814089
El paciente no esta identificado :(, esto genera problemas
El paciente es militar y redimio el codigo MILITAR, que le da un 50% de descuento en su cirugia B)
```

Opciones de identificacion del paciente

Si el paciente esta identificado se aparta un quirofano y se busca un doctor de manera urgente(no esta garantizado el exito de esto) y se reserva tal quirofano y tal doctor para comenzar la operacion, sino se consiguio doctor y quirofano el paciente se muere, en el caso de que haya cirugia al finalizar se genera la factura, considere que el valor de la factura depende de la especialidad del doctor que a su vez depende de la razon por la que llego al hospital

Si el paciente no esta identificado, la ejecucion es especial, donde trata de escoger un doctor de la especialidad que necesita el paciente para operar, si no es posible entonces el paciente se muere

```
El doctor Medico de planta Juana de la especialidad cardiolog@ justo esta pasando y es muy buena gente
Segun las politicas del hospital, no podemos operar al paciente al no estar identificado
Pero... podriamos operarlo de todas maneras  (ಠ ʖ ಠ)
Puedes decir por el doctor y decidir si operar al paciente,
INGRESA ;) para operar, de lo contrario, entedere tu señal como que no quieres operar al paciente
```

Ejemplo de ejecucion(Paciente no identificado)

Si se da la señal para operar al paciente se necesita ademas escoger un quirofano libre, si se escoge un quirofano ocupado, entonces descubren este inclumplimiento a las normas por lo que despiden al doctor y el paciente muere, al terminar la cirugia, se genera una factura, el valor de la factura depende de la especialidad del doctor que a su vez depende de la razon por la que llego al hospital

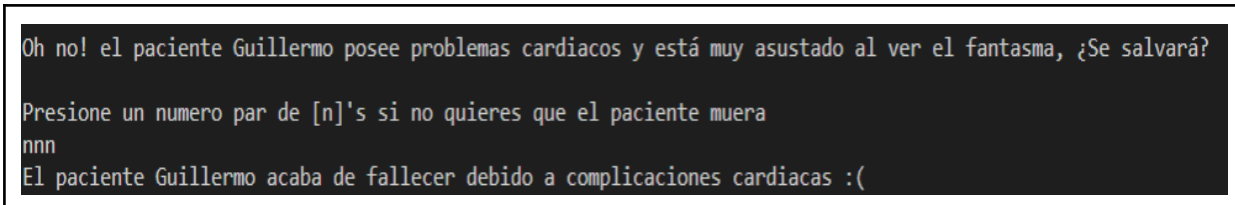
¡Peligro! Aquí duerme fantasmon

Esta funcionalidad hace que ciertas instancias de tipo pacientes que ya fueron iniciadas dejen de existir dependiendo de si poseen problemas cardiacos, es por eso que al ejecutar esta función pueden derivar distintos resultados que mostraremos a continuación.

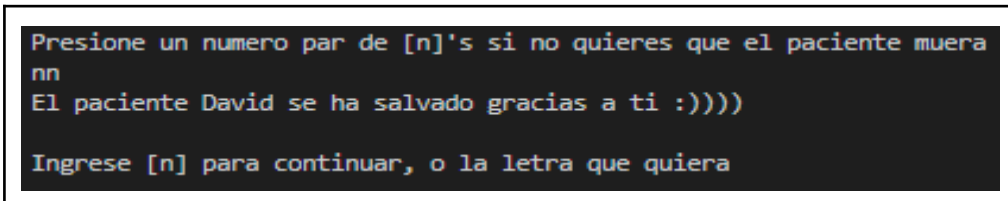
Objetos que intervienen en la funcionalidad:

- Un hospital
- Varios pacientes
- Varios quirófanos

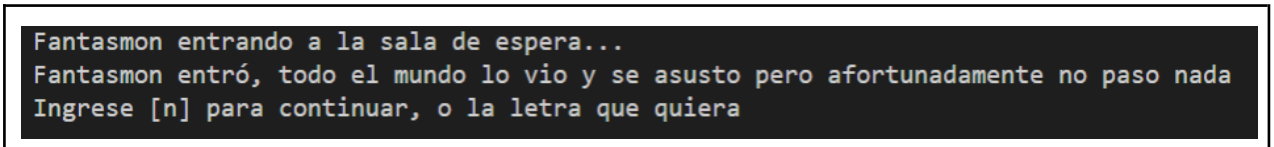
Resultados que muestra al usuario:



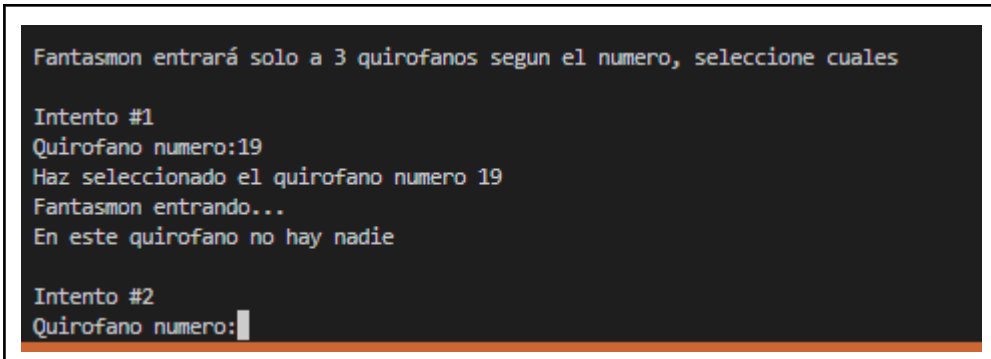
Posible resultado



Posible resultado



Posible resultado



Posible resultado

Proyecto Experimental

En este hospital tan particular tiene un proyecto experimental para curar una gripa operando, como es un proyecto experimental se necesitan doctores que esten dispuestos a ello, y el usuario tiene el poder de elegirlos, podras elegir 3 diferentes doctores, o si el hospital tiene menos doctores, seran los que se pueda

```
Este hospital es pionero en investigacion!
Se ha investigado de una manera en la que se puede curar la gripa
Para ello vas a tener que elegir a ciertos doctores para que hagan una operacion
Escoje cuales doctores van a hacer la operacion experimental
Son 3 maximo, si hay menos de 3 doctores, entonces seran esos
[0] Medico de planta Juana de la especialidad cardiolog@
[1] Medico de planta Rebeca de la especialidad cirugan@ plastic@
[2] Jefe Derek de la especialidad neurocirujano
[3] Medico de planta Cristina de la especialidad cardiolog@
```

Ejemplo de ejecucion

Cuando se ha elegido doctores, por cada uno hace el siguiente procedimiento, se crea un tablero donde las dimensiones dependen de la jerarquia del doctor, y la idea es completar un minijuego bastante parecido al buscaminas
Si usa bisturi en una celula mala(similar a una mina) entonces esta eliminando esa celula mala, si hace succion en una celula mala entonces esto hace que la cirugia sea un fracaso y se mata al paciente

```
Medico de planta Juana de la especialidad cardiolog@ va a comenzar su operacion
Recuerda que operar a alguien es como jugar buscaminas, solo no debes dar donde no es
El tablero es de 10x10
Por lo que las filas y columnas van desde 1 hasta 10
Si escoges fuera de este rango puedes dañar el sistema!
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
Bisturi o succion?
[1] Succion | [2] Bisturi
Elije instrumento:
```

Ejemplo de ejecucion

El hospital es bastante estricto, por lo que si la cirugía es un fracaso, se despide al doctor

Ayuda a un Doctor en apuros

En esta funcionalidad ascendemos a un doctor que lo seleccionará el usuario, hay unos rangos establecidos:INTERNO, PLANTA, DIRECTORMEDICO Y JEFE en orden ascendente, para esto hay un pequeño juego que dependiendo si lo gana el usuario, el doctor que previamente se selecciona asciende a un mejor cargo y en caso de perderlo este se mantiene en su cargo.

Objetos que intervienen en la funcionalidad:

-Un Hospital
-Varios Doctores
-Varios Quirófanos

```
En el hospital hay doctores ocupados que están en los quirofanos y otros que estan disponibles
seleccione a cuales quieres ayudar:
[1] Doctores que estan en los quirofanos
[2] Doctores que estan disponibles en el hospital

1
Haz seleccionado a los doctores que están trabajando en los quirofanos
Estos son los doctores:
[1] Doctor/a Jaime      [2] Doctor/a Felipe

Seleccione el doctor/a que quieras ayudar:1

Haz seleccionado al doctor/a Jaime y el cargo que ocupa es de PLANTA
```

Ejemplo de ejecución

```
El siguiente juego se llama ahorcadito
Adivina la palabra!
_ _ _ _
Introduce una letra:
h
h _ _ _
Introduce una letra:
o
h o _ _
Introduce una letra:
l
h o l _
Introduce una letra:
a

Felicidades!! has acertado la palabra: h o l a
El/la doctor/a Jaime ahora ocupará el cargo de DIRECTORMEDICO y tendrá un aumento
```

Posible resultado

```
Seleccione el doctor/a que quieras ayudar:5
Este doctor ya no hace parte del hospital
Ingrese [n] para continuar, o la letra que quiera
```

Posible resultado

Posible resultado

Manual de usuario con los elementos necesarios para poder evaluar el correcto funcionamiento del sistema

Experimento sale mal

La ejecucion de la funcionalidad guia al usuario, cuando aparezca el siguiente mensaje

```
Ingrese [n] para continuar, o la letra que quiera
```

Ingrese n o cualquier otra letra y presionar enter para que la funcionalidad siga su curso

---

Paciente urgente necesita ayuda

La ejecucion de la funcionalidad guia al usuario, cuando aparezca el siguiente mensaje

```
Ingrese [n] para continuar, o la letra que quiera
```

Ingrese n o cualquier otra letra y presionar enter para que la funcionalidad siga su curso

La ejecución de esta funcionalidad tiene varias decisiones aleatorias, por lo que la interaccion con el usuario no esta determinada y se presentan las siguientes posibilidades

En caso de que el paciente sea no identificado, se tienen 2 posibilidades

```
Segun las politicas del hospital, no podemos operar al paciente al no estar identificado
Pero... podriamos operarlo de todas maneras  (ʘ‿ʘ)
Puedes decir por el doctor y decidir si operar al paciente,
INGRESA ;) para operar, de lo contrario, entedere tu señal como que no quieres operar al paciente
```

Si se quiere operar al paciente, se le debe dar como input al programa “;)”, sin espacios antes ni despues, y oprimir enter

Despues para elegir un quirofano, el usuario debe dar como input al programa un numero entero, que este dentro de los que se imprime en pantalla

```
Vamos a conseguir
[0] Quirofano con el doctor Jaime que esta operando al paciente Esteban
[1] null
[2] null
[3] Quirofano con el doctor Felipe que esta operando al paciente Juanito
[4] null
[5] null
```

¡Peligro! Aquí duerme fantasmon

La ejecución de la funcionalidad guia al usuario, cuando aparezca el siguiente mensaje

```
Ingrese [n] para continuar, o la letra que quiera
```

Ingrese n o cualquier otra letra y presionar enter para que la funcionalidad siga su curso

---

Operacion Experimental

Para el correcto funcionamiento del programa, se espera que el usuario entre los valores adecuados al programa en cualquiera de las siguientes preguntas

- `Escojo a :` se imprime junto con una lista de doctores con un numero asignado, se espera que el usuario entregue al programa uno de los numeros enteros(sin decimales) al programa, no se vale repetir numeros
- `Elije instrumento:` Se necesita que el usuario ingrese ya sea el 1 o 2 al programa
- `Fila(min:1,max:10) :` (considere que el valor maximo puede variar), el programa espera un entero entre el minimo y maximo valor que son imprimidos en pantallas
- `Columna(min:1,max:10) :` (considere que el valor maximo puede variar), el programa espera un entero entre el minimo y maximo valor que son imprimidos en pantalla

---

Ayuda a un Doctor en apuros

La ejecución de la funcionalidad guia al usuario, cuando aparezca el siguiente mensaje

```
Ingrese [n] para continuar, o la letra que quiera
```

Ingrese n o cualquier otra letra y presionar enter para que la funcionalidad siga su curso