Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

GENERADOR Y ANALIZADOR DE MUESTRAS POR PACIENTE

Carnet 201445840 – Luis Humberto Lémus Pérez

Resumen

La presente documentación describe las distintas funcionalidades, métodos, y validaciones que se utilizaron para la realización de la aplicación, la aplicación tienen como objetivo el análisis y la generación de nuevas muestras a partir de coordenadas con las células infectadas que se obtienen de un archivo .xml; en el cual se lee obteniendo no solo los datos de infección sino también la información de cada paciente como su nombre, edad , el tamaño de su muestra y cuantos periodos se debe calcular.

Palabras clave

Paradigma
Nodo
Lista simple enlazada
Metodos
Clases
Interfaz de usuario
Polimorfismo
Modularidad
Herencia

Abstract

This documentation describes the different functionalities, methods, and validations that were used to carry out the application, the application aims to analyze and generate new samples from coordinates with the infected cells that are obtained from a file .xml; in which it is read obtaining not only the infection data but also the information of each patient such as their name, age, the size of their sample and how many periods should be calculated.

Keywords

Paradigm

Node

Single Linked List

Methods

Classes

User Interface

Polymorphism

Modularity

Inheritance

Introducción

Describe las funciones y que paradigma de programación se utilizó así como la lógica utilizada para la solución del problema también se indica los diagramas de clases y se explica la utilización de listas enlazadas para guardar la información requerida.

Se Describe las validaciones utilizadas para mostrar los errores que pudiesen ocurrir así como la utilización de la aplicación.

Desarrollo del tema

a. Explicación de nodos y listas simples

Nodo: un nodo contiene un dato y un apuntador al siguiente nodo

*Figura 1.*Nodo de una lista simple.

Lista simple:

Una lista es una secuencia de elementos del mismo tipo o clase almacenados en memoria. Las listas son estructuras lineales, donde cada elemento de la lista, excepto el primero, tiene un único predecesor y cada elemento de la lista, excepto el último, tiene un único sucesor

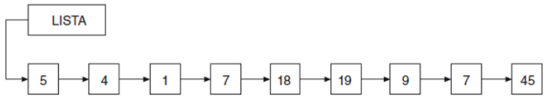


*Figura 2.*representacion de una lista simple.

b. Paradigma de programación utilizado

Programación orientada a objetos:

Polimorfismo
Modularidad
Herencia

Paradigma Imperativo

Programación modular

c. Diagrama de clases

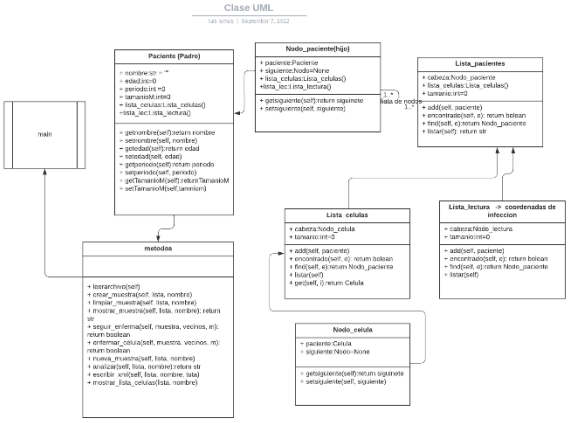


Figura 3.Diagrama de clases.

d. Lógica de cálculo de vecinos de una célula

	0	1	2	3	4
0					
1		v6	v7	v8	
2		v4	celula	v5	
3		v1	v2	v3	
4					

Figura 4. Representación de celula.

Dada una muestra de 5*5 se muestra una célula de e coordenadas (2,2), el cálculo de sus vecinos viene dado por:

```
vecino1=Vecino(self.x-1, self.y+1)
vecino2=Vecino(self.x-0, self.y+1)
vecino3=Vecino(self.x+1, self.y+1)
vecino4=Vecino(self.x-1, self.y+0)
vecino5=Vecino(self.x+1, self.y+0)
vecino6=Vecino(self.x-1, self.y-1)
vecino7=Vecino(self.x-0, self.y-1)
vecino8=Vecino(self.x+1, self.y-1)
```

e. Métodos Principales utilizados

```
def add(self, paciente):

    recibido = Nodo_paciente(paciente)  #convertimos el nodo que se trae en un nodoPer
    nuevo = Nodo_paciente(recibido.paciente)
    nuevo.lista_celulas=recibido.lista_celulas
    nuevo.lista_le=recibido.lista_le

    if self.esVacia():

        self.cabeza = recibido
        messagebox.showinfo("info", "Se agrego correctamente el paciente")
        self.tamanio=self.tamanio+1
    elif self.encontrado(recibido.paciente.nombre) == False :

        aux = self.cabeza
        while (aux.getSiguiente() != None) :
            aux = aux.getSiguiente()

        aux.setSiguiente(nuevo)
        messagebox.showinfo("info", "Se agrego correctamente el paciente")
        self.tamanio= self.tamanio+1
```

Figura 5. Método add de una lista simple

```
def mostrar_muestra(lista, nombre):

    encontrado:'Nodo_paciente'
    encontrado=lista.find(nombre)
    if encontrado != None:
            tam= encontrado.lista_celulas.tamanio
            m=encontrado.paciente.tamanioM
            grafico=""
            for i in range(tam):
                    cel=encontrado.lista_celulas.get(i)
                    reprentacion="|"
                    salto='\n'

                    if cel.enferma:
                        reprentacion="|"
                    if cel.x==m-1:

                        grafico=grafico + reprentacion+salto
                    else:
                        grafico=grafico+reprentacion
            return grafico
```

Figurara 6. Método mostrar muestra por paciente

```
def crear_muestra(lista, nombre):
        m=0

        encontrado:'Nodo_paciente'
        encontrado=lista.find(nombre)
        if encontrado != None:

            m=encontrado.paciente.tamanioM

            zize=0

            for  y in range(0,m):
                for x in range(0,m):

                    encontrado.lista_celulas.add(Celula(False, x, y, zize))
                    zize=zize+1

            ran=encontrado.lista_le.tamanio

            for  i in range(ran):
                clase=encontrado.lista_le.get(i)
                celula=encontrado.lista_celulas.find(clase.x, clase.y)
                celula.enfermar()
```

Figurara 7. Método crea la muestra por paciente

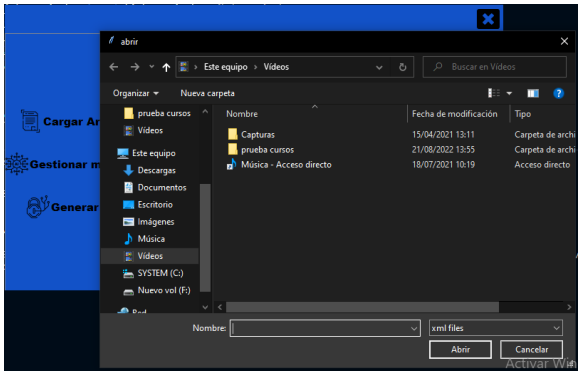
```
def analizar(lista, nombre):

    metodos.nuevaMuestra(lista,nombre)
    cadena=metodos.mostrar_muestra(lista, nombre)
    return cadena
```

Figurara 7. Método analiza la muestra por paciente

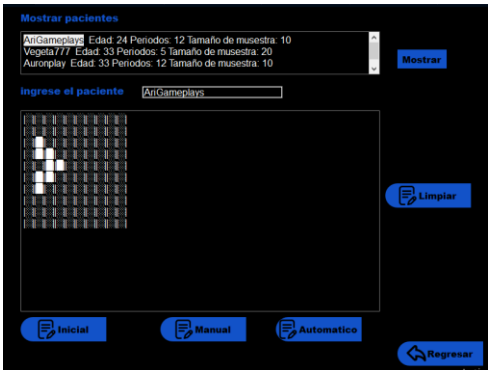
f. Interfaces de usuario

Figurara 8.interfaz principal



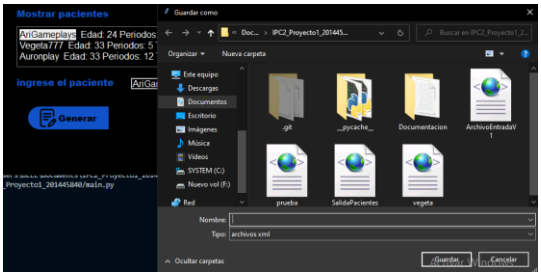
Figurara 8.seleccion de archivo xml

Figurara 9.gestion de muestras inicial por paciente



Figurara 10.gestion de muestras manual periodo por periodo

Figurara 11.gestion automática todos los periodos por usuario



Figurara 12.genaracion de xml por paciente

Conclusiones

El programa es capaz de a través de un archivo de entrada xml guardar los datos del paciente en listas simples y dentro de ellas también guardar su muestra inicial dada, y poder generar mediante sus periodos el avance de la enfermedad, de forma manual (periodo por periodo) o de forma automática(todos los periodos dados).

También es capaz de generar un xml de salida por cada usuario indicando el avance de la enfermedad si es leve o si por el contrario es grave, mortal.