Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructuras de datos

Titulo: Manual técnico, proyecto Fase 1

Nombre: Xhunik Nikol Miguel Mutzutz

Carnet: 201900462

Manual técnico – Fase 1

Objetivos

Objetivo general

Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de una aplicación que permita manipular la información de forma óptima

Objetivos específicos

- Demostrar los conocimientos adquiridos sobre estructuras de datos lineales poniéndolos en práctica en una aplicación de simulación.
- Utilizar el lenguaje Java para implementar estructuras de datos lineales.
- Utilizar la herramienta Graphviz para graficar estructuras de datos lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación.

Descripción

La empresa “Drawing Paper” se dedica a imprimir imágenes en distinto tamaños y tipos de
papel, actualmente ha tenido un crecimiento exponencial en la cantidad de clientes, lo cual
le ha generado problemas en el área de recepción de pedidos, por lo cual se le solicita a
usted como un estudiante de ingeniería en sistemas que aplique sus conocimientos en
estructuras de datos para solucionarlos.
Deberá desarrollar una aplicación utilizando las estructuras de datos y sus algoritmos
explicados en clase, de tal forma que pueda simular los diferentes procesos que se dan en
la empresa. Dicha aplicación deberá ser capaz de representar las estructuras de forma
visual, mediante la utilización de bibliotecas soportadas (Graphviz).

Simulación

- La empresa desea realizar una simulación de todo el proceso que conlleva imprimir una imagen, es decir, desde que los clientes realizan la solicitud de imprenta, hasta que dicha imagen se les entrega impresa.
- El tiempo de simulación se medirá mediante pasos, es decir debe existir una opción en el menú para avanzar un paso.
- El funcionamiento en general de la empresa es el siguiente:
- La empresa posee un número “n” de ventanillas de recepción donde son tomados
- los pedidos, este valor se debe definir al inicio.

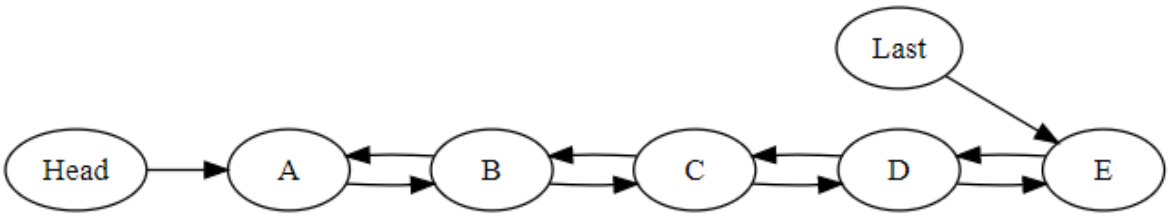
- Los clientes al llegar a la empresa ingresan a una cola de recepción, salen de dicha cola en el momento en que una ventanilla esté disponible.
- Cuando un cliente ingresa a una ventanilla se almacena su información en una lista.
- La cantidad de pasos que tarda un cliente en la ventanilla es equivalente a la cantidad de imágenes que desea imprimir.
- La cantidad de clientes que llegan por paso deberá ser aleatoria en un rango de 0 a 3.
- Una vez es atendido el cliente en la ventanilla, él pasará a una lista de clientes en espera y sus imágenes pasarán a una cola de impresión dependiendo de su tipo (color, blanco y negro).
- Existen dos impresoras, una a color y otra en blanco y negro, cada una con sus respectivas colas de impresión.
- La impresora en blanco y negro tarda 1 paso en imprimir una imagen.
- La impresora a color tarda 2 pasos en imprimir una imagen.
- Cuando una imagen termina de ser impresa es ingresada a la lista de imágenes de cada cliente que se encuentra en la lista de espera, como se especifica en el punto 5.
- El cliente sale de la lista de espera cuando todas sus imágenes son impresas.
- Cuando un cliente sale de la empresa se actualiza en su registro la cantidad de pasos que estuvo esperando su imagen.

Estructuras implementadas

Lista enlazada Simple

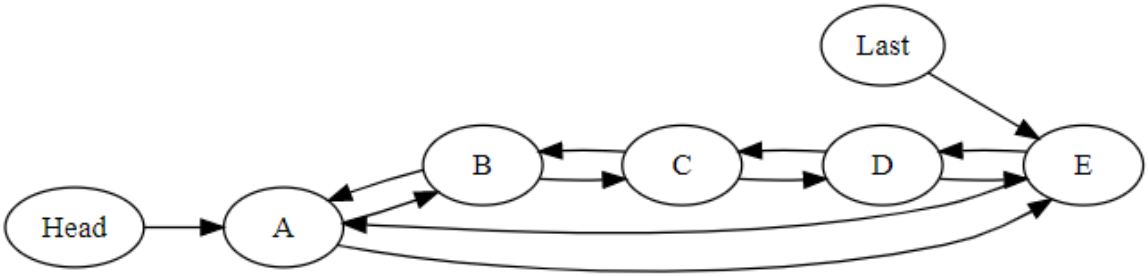
- Esta lista contiene un nodo head que apunta al primer elemento y cada elemento apunta al nodo sucesivo hasta el ultimo el cual apunta a NULL.

Lista enlazada doble



- Esta lista contiene un nodo head que apunta al primer elemento y un nodo last que apunta al ultimo, cada elemento contiene una referencia al nodo siguiente y al nodo anterior excepto el primero y el ultimo los cuales tienen un apuntador a NULL hacia el anterior y el siguiente respectivamente.

Lista Circular Doble



- Esta lista se caracteriza por que tanto el ultimo elemento como el primero están enlazados mutuamente de manera que se puede recorrer la lista en cualquier sentido e infinitas veces llegando siempre a los mismos elementos.

Cola

- La cola esta implementada heredando la lista enlazada doble, en la cual solo se exponen los métodos de agregar al inicio y eliminar al final, de manera que implementa la estructura FIFO.

Pila

- La pila esta implementada heredando de la lista enlazada simple ya que solo es necesario manipular el head.

Modelos

Ventanilla

```java
public class Ventanilla {
    public String id;
    public String nombre;
    public Cliente cliente;
    public PilaImpresiones pilaImagenes;

    public Ventanilla(String nombre) {
        this.id = EddProyectoFase1.generateGuid();
        this.nombre = nombre;
        this.pilaImagenes = new PilaImpresiones();
    }

}
```

Impresora

```java
public class Impresora {
    public String id;
    public TipoImagen tipo;
    public ColaImpresoras queue;
//    public int turnos;

    public Impresora(TipoImagen tipo){
        this.id = EddProyectoFase1.generateGuid();
        this.tipo = tipo;
        this.queue = new ColaImpresoras();
//        this.turnos = 0;
    }

    public void addImage(Imagen img){
        this.queue.enqueue(img);
    }

    public void printImage(ListaClientesEspera espera){
        Imagen img = this.queue.dequeue();
        if (img != null){
            Cliente cl = espera.getById(img.clientId);
            espera.appendImageToCliente(img.clientId, img);
            System.out.println("Imprimiendo imagen: "+ img.tipo.toString() + " del cliente: "+cl.nombre);

        }
    }
}
```

Imagen

```java
public class Imagen {
    public String id;
    public TipoImagen tipo;
    public String clientId;

    public Imagen(TipoImagen tipo) {
        this.id = EddProyectoFase1.generateGuid();
        this.tipo = tipo;
    }

    public Imagen(TipoImagen tipo, String clientId) {
        this(tipo);
        this.clientId = clientId;
    }
}
```

Cliente

```java
public class Cliente {
    public String id;
    public String nombre;
    public int imgColor;
    public int imgBlancoNegro;
    public int pasos;
    public String idVentanilla;
    public int imgBlancoNegroConst;
    public int imgColorConst;
    public ListaImpresiones listaImages;

    public Cliente(){
        this.id = EddProyectoFase1.generateGuid();
        this.listaImages = new ListaImpresiones();
        this.pasos = 0;
    }

    public Cliente(String nombre) {
        this();
        this.nombre = nombre;
    }

    public Cliente(String nombre, int imgColor, int imgBlancoNegro) {
        this(nombre);
        this.imgColorConst = this.imgColor = imgColor;
        this.imgBlancoNegroConst = this.imgBlancoNegro = imgBlancoNegro;
    }

}
```

Métodos

Insertar al Inicio

Implementado en: Lista simple, lista doble, por extensión a Cola y Pila

Inserta un elemento antes del head actual y mueve el apuntador head al nuevo elemento

Insertar al final

Implementado en: Lista simple, lista doble, lista circular doble

Inserta un elemento al final y si existe un apuntador ultimo lo desplaza al nuevo elemento

Obtener primero

Implementado en: Lista simple, lista doble, por extensión a Cola y Pila

Obtiene el primer elemento sin alterarlo.

Obtener ultimo

Implementado en: Lista doble, por extensión a Cola

Obtiene el último elemento sin alterarlo.

Eliminar primero

Implementado en: Lista simple, lista doble, por extensión a cola y Pila

Elimina el primer elemento y mueve el head al siguiente.

Eliminar ultimo

Implementado en: Lista doble, por extensión a Cola

Elimina el ultimo elemento y mueve el nodo ultimo al anterior.

Verificar si vacío

Implementado en: Todas

Verifica si existe algún elemento.

Limpiar

Implementado en: Todas

Elimina todos los elementos de la lista.

Buscar

Implementado en: Lista circular doble

Navega hasta encontrar un elemento con una propiedad que coincida con el parámetro, luego retorna.

Modificar

Implementado en: Lista circular doble

Navega hasta encontrar un elemento que coincida con el parámetro y almacena la nueva información en esa posición.