INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPÚTO

PROGRAMACIÓN ORIENTADA A OBJETOS

PRÁCTICA 2: MOSAICO DE BOTONES

PROF. ROBERTO TECLA PARRA

GRUPO 2CM4

ZEPEDA FLORES ALEJANDRO DE JESÚS

20/03/3018

**OBJETIVO**

Desarrollar un Applet en lenguaje de programación Java con 40 botones, en cada botón se incluirá una imagen. Mostrar en una etiqueta, la imagen que se encuentre en el botón donde el usuario dio click.

**INTRODUCCIÓN**

La clase BufferedImage se usa para mantener una representación de una imagen en memoria dentro de una aplicación Java, de modo que la puedes modificar o guardar en cualquiera de los formatos estándar propios de imágenes. Esta clase será de mucha importancia, ya que, gracias a esta, podremos leer un número finito de imágenes contenidas dentro de la misma carpeta que el programa.

**DESARROLLO**

Para poder desarrollar la práctica, primero necesitamos establecer las clases que vamos a utilizar para el funcionamiento óptimo de nuestro programa, para este caso son las siguientes:

import java.awt.event.\*;

import javax.swing.\*;

import java.applet.\*;

import java.awt.\*;

import javax.imageio.\*;

import java.io.\*;

import java.awt.image.\*;

Después de establecer nuestras clases, declaramos una clase pública a la cual llamaremos ***Mosaico***, la cual extiende una clase Applet e implementa la interfaz ***ActionListener***.

public class Mosaico extends JFrame implements ActionListener

JButton botones[] = new JButton[40];

BufferedImage imagenes[];

String s= ""; Container c;

JLabel mensaje;

El siguiente paso será inicializar el Applet, mediante el constructor; la función de este método es reservar memoria y llamar al constructor de la clase utilizando el operador new, seguido del nombre del objeto o componente, logrando así el objetivo. Posteriormente, agregaremos los componentes y el addActionListener.

public Mosaico(){

c=getContentPane();

imagenes = new BufferedImage[40];

setLayout(new GridLayout(8,5));

for(int i = 0; i < 40; i++){

s= i+ ".jpg";

imagenes[i] = leeImagen(s);

mensaje = new JLabel(" ");

c.add("South",mensaje);

botones[i].addActionListener(this);

}

}

El paso siguiente será implementar el método actionPerfomed (ActionEvent e) el cual nos ayudara a poder manipular nuestros objetos por medio de eventos y estos se activarán.

public void actionPerformed ( ActionEvent e ){

JButton b=(JButton)e.getSource();

mensaje.setIcon( b.getIcon() );

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

Para este Applet, utilizaremos un nuevo método BufferedImage, la finalidad de este método es obtener una representación de una imagen en memoria dentro de una aplicación.

public BufferedImage leeImagen(String nombre){

BufferedImage imagen;

try{

imagen = ImageIO.read(new File(nombre));

}catch (IOException e){

System.err.println(e+" "+nombre);

return null;

}

return imagen;

}

**CONCLUSIÓN**

La importancia de esta práctica, recae en la utilización de arreglo de imágenes y botones, además de la utilización del método BufferedImage, este simplifica y optimiza el código, de lo contrario, se tendría que llenar los arreglos posición por posición.