Proyecto POO “Calicua”

Daniel Felipe Agudelo, Julian Serna Huertas.
Universidad Nacional de Colombia - sede Bogotá
Fecha de entrega del informe: 2 de Diciembre de 2022

Resumen - Para la materia de programación orientada objetivo se desarrolló un programa que organice uml enfocado en el desarrollo de uml para la plataforma de arduino, por la naturaleza de la misma plataforma arduino, se busca un enfoque educativo para aquellos que no tienen experiencia ni en arduino ni en uml.

- Introducción

En el siglo XXI, las competencias en el desarrollo de tecnologías y en la innovación se necesitan herramientas para aprender en distintas plataformas como lo son arduino, java, UML y metodologías dinámicas como SCRUM.

Muchas tecnologías se han desarrollado para motivar a la gente a aprender no únicamente a programar, se busca que la tecnología influya en la vida de las personas de manera positiva, llevando la innovación y la capacidad de desarrollar soluciones tecnológicas a todo el mundo sin importar su raza o condición social. Es el marco de la democratización del conocimiento y la tecnología donde se crea Caliqua, un software que pretende aportar una plataforma de desarrollo UML integrada con arduino, para el desarrollo de proyectos de electrónica y mecatrónica de forma fácil y organizada.

- Tecnologías asociadas

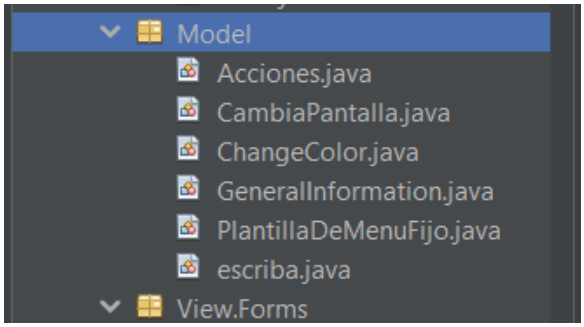
Se utilizó arduino, el IDE para desarrollo de java Netbeans, y de referencia el manual de UML,al momento de desarrollar el prototipo, utilizamos librerías de Google y el framework de arduino para el desarrollo de software.

- Diseño del prototipo de software

Se diseñó el software con el siguiente algoritmo:

1. El usuario crea un diagrama UMl
2. el usuario va desarrollando el UML de su proyecto, casos de uso, clases, etc.
3. tras esto, algunos diagramas de proceso podrán ser convertidos a código de Arduino

- Explicación detallada del código generado.



El paquete Model posee toda la información de background que no se ve reflejada directamente en el programa, se trata de información en el código, otro paquete se encarga de revelar esa información.

La clase Acciones es un diccionario que posee las traducciones del texto introducido por el usuario a código:

```
package Model;

public class Acciones {
  //cada método es una traducción.
  public String decirSalida(String n){

    String s = "PinMode("+n+")";

    return s;
  }

  public String tiempo(String n){

    String s = "delay("+n+")";

    return s;
  }

  public String darEnergia(String n, String n2){

    String s = "digitalWrite(" + n + "," + n2 + ")";

    return s;
  }

  public String verificarPin(String n,String m){
```

```
    String s = m + " = digitalRead (" + n + ")";

    return s;
  }

  public String definir(String n){
    String s = "#define " + n;

    return s;
  }

  public String mientrasQue(String n){
    String s = "for ("+n+") { ";

    return s;
  }

  public String acabar(){
    String s = "}";

    return s;
  }
  public String igualar(String n,String m){
    String s = n + " = " + m;

    return s;
  }


}
```

CambiaPantalla: como su nombre lo dice, cambia la pantalla dependiendo del botón que el usuario elija, en resumen cada menú, las barras al lado derecho e izquierdo del programa, es un jList custom que usando un evento permite generar un número dependiendo de cuál ítem del tipo MENU sea elegido, lo del tipo se explicará más adelante, ahora bien, usando el método set form se borra el anterior del contenedor designado, en este caso un jpanel, luego se añade y se refresca la página. El método change utiliza instancias de cada form o pantalla que el usuario ve, el evento anteriormente mencionado y posibilidades de elección dependiendo del número que de el evento para ejecutar el método setform y ejecutar cualquier otra línea de código necesaria para el funcionamiento del programa como otros métodos, de esa manera emulamos un jtabbedPane pero sin sus problemas y preconfiguración, que nos pareció fea, el método change tambien posee como parametros que menus son lo que ejecutarán las opciones y los contenedores a los que irá la acción.

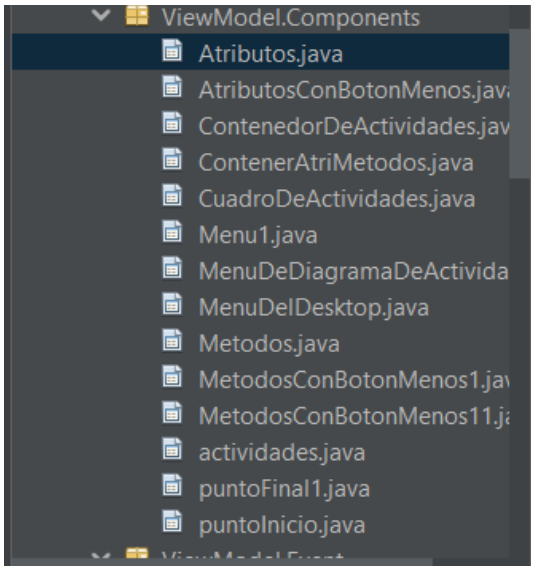
Para evitar que el informe sea demasiado largo por favor visualice el código directamente del rar en el GitHub.

ChangeColor: como su nombre lo indica , solamente cambia el color del programa, para hacer eso, usa el método cambiador, cada componente que se muestra en el programa posee un método superpuesto llamado paint children que crea figuras cancelando el graphics original de los componentes e introduciendo la figura que quieras, de esa manera hacemos los degradados, flechas, etc. El tema es que cambiador solamente cambia la variable tipo de todos estos lo que repinta dependiendo del número que tenga este atributo.

GeneralInformatión: guarda la información de las instancias de los componentes generados por CambiaPantalla, todo lo que se les cambie se guarda aquí, hace parte del sistema de guardado, aunque al estar incompleto esta clase realmente no funciona.

PlantillaDeMenuFijo: genera los atributos del componente que formará los elementos del jList mencionado anteriormente, este genera la imagen llamada Icon un nombre que realmente no es usado por un cambio en el diseño y un tipo, que ya se vio una función anteriormente, pero que por ejemplo si es de tipo EMPTY, será como si no existiera pero será tomado en cuenta por el jList y el evento mencionado anteriormente.

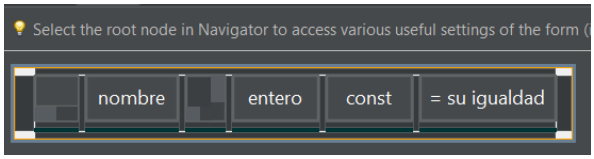
Escriba: escribe un String específico en un archivo TXT.



El paquete ViewModel.Componets hace parte del ViewModel, o sea, la lógica detrás de la creación del View, lo que se le muestra al usuario.

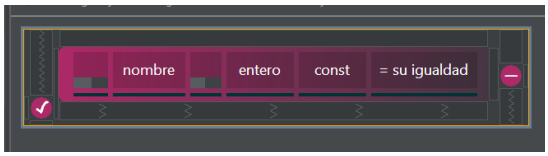
Estos todos son componentes custom que cumplen su función tal cual las piezas de un rompecabezas forman una pintura, son como los jcomponents normales pero hechos a la medida para que se vean y actúen de una manera específica de acuerdo a nuestras necesidades, estos por lo general se conforman de muchos jcomponents y su paintChildren está sobreescrito para crear imágenes con Graphics 2D tal cual un círculo, flecha, fondo degradado, curvatura en un cuadro, etc, etc.

Atributos:



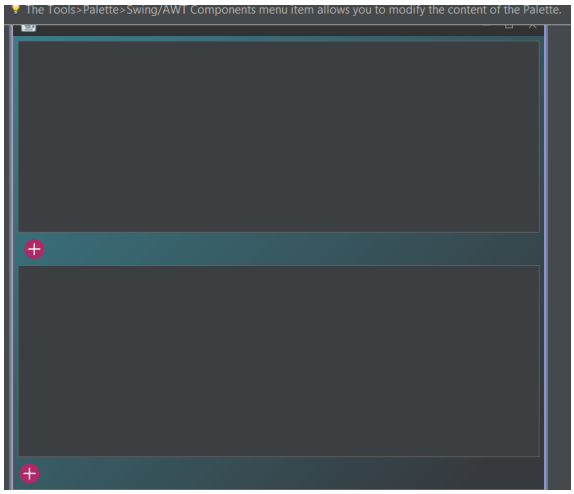
Es la plantilla de los atributos, es un jpanel con paintChildren sobrescrito a degradado de violetas para contraste con el azul característico de la versión light y dark del programa, usa la librería swingx para crear un autocompletado de los espacios más importantes mostrados en la imagen, y para crear el código se usa su método añadir donde usa el diccionario anteriormente mencionado para que dependiendo de lo que se escriba se genere un código en específico.

AtributosConBotonesMenos:



Usa la clase hecha antes como componente customs y con jlabel con icono genera esos botones de menos y chulo que se ven, en esta clase se dan los eventos de los botones solamente, chulo añade al array principal en main y menos borra del array y borra por completo al componente custom AtributosConBotonesMenos, su paintChilden también está sobrescrito a los colores de los menús.

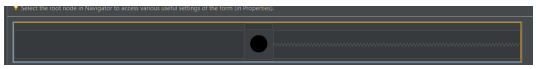
ContenerAtriMetodos:



Se encarga de ser el jinternalFrame que funge como contenedor de la clase anterior, en el jpanel de arriba se almacenan los atributos, abajo los métodos que se crean igual que los atributos pero con datos distintos y usan a las clases Métodos análogo de Atributos y MetodosConBotonesMenos1 analogo de AtributosConBotonesMenos y de la misma forma funcionan los uml de actividades pero con actividades análogo de Atributos y MetodosConBotonesMenos11 analogo de MetodosConBotonesMenos1 pero estos últimos diagramas son contenidos en Contenedor de Actividades, con la diferencia de que Actividades primero se usa como componente custom de CuadroDeActividades que tiene un paintChildren sobrescrito para crear la forma del cuadro de actividades de UML

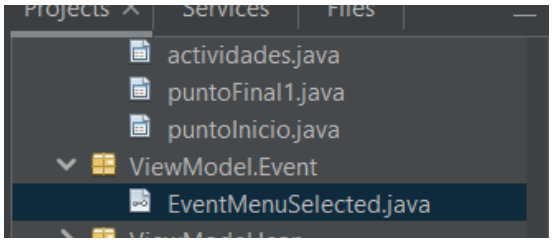
puntoFinal1:

y puntoInicio:



Son jpanels con jlabel con icono capaces de emular las formas iniciales y finales de un diagrama de actividades UML.

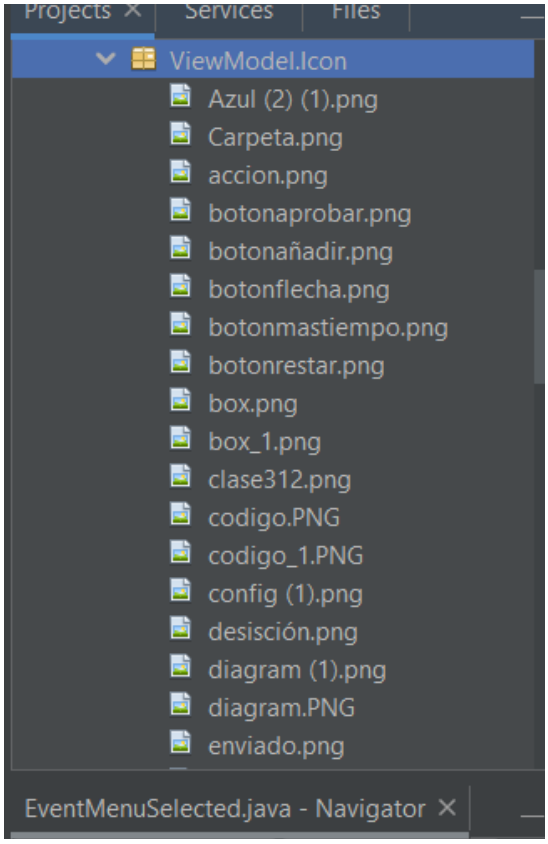
De las clases menú se hablará más tarde.



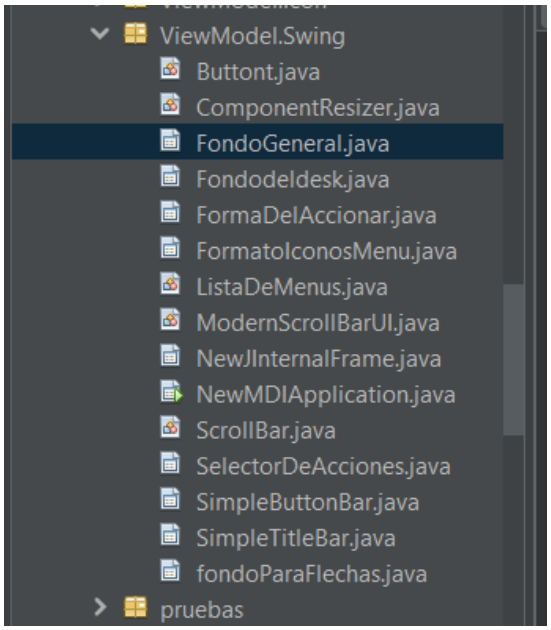
El paquete ViewModel.Event sólo puede contener interfaces o eventos como se refirió antes, me referiré de esas maneras a las interfaces no gráficas.

```
package ViewModel.Event;

public interface EventMenuSelected {

    public void selected(int index);

}
```

La única clase del paquete permite generar el contador del cual se habló en Cambia pantalla.



El paquete ViewModel.Icon posee todos los iconos en formato png para transparencia de imagen y no tener que hacer todo los visual en Graphics 2D.



El paquete ViewModel.Swing contiene jcomponent poco modificados o de librerías que por contraparte a components solo poseen un tipo de jcomponent.

Buttont, ComponentResizer, ModernScroolBarUI, ScrollBar, SimpleButtonBar, SimpleTitleBar hace parte de una librería de forms nuevos, por lo que son jcomponents de la librería que se modificaron un poco para el proyecto.

FondoGeneral: es un jpanel con paintChildren sobrescrito y como se dice es el fondo de nuestro programa.

FormadelAccionar: es el cuadro curvo de los cuadros de acciones, solo la forma por lo que es un jpanel con paintChildren sobrescrito.

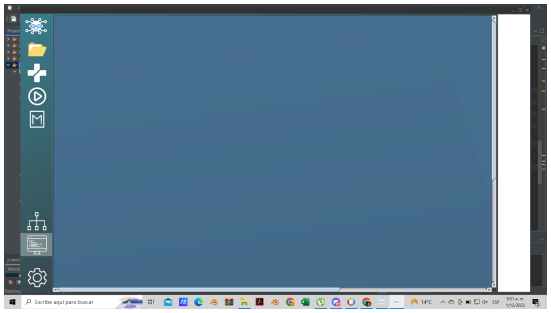
FormatoDeIconos: es la excepción porque se compone de más componentes pero solo es la plantilla que se usará más adelante para crear el jlist de los menos, del cual se hablará más adelante.

ListaDeMenus: aqui se crea el formato del jList, en este crean los eventos de seleccion y cambio de color si el mismo, se crea un nuevo model para jList donde se usa el formato de la clase anterior y se construye un metodo que permite añadir facilmente contenido a los Jlist que es el model dependiendo de la imagen, texto, tipo, si es MENU, EMPTY o TITLE, para generar el model se sobrescribe el renderizador de celdas del array de jlist, asi se creo el model que después fue usado como componente custom para los Menu clases que

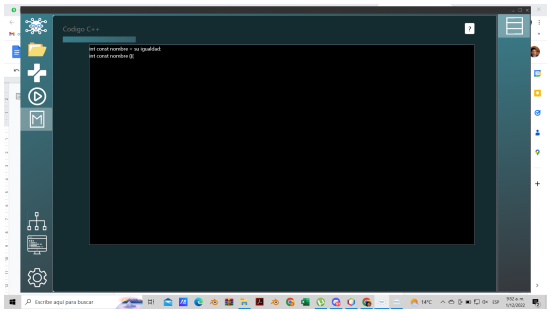
postergamos y justo porque se queria explicar el model tan raro que poseen.

Y fondoParaFlechas es una clase no terminada que generará flechas renderizadas como uniones entre los componentes.

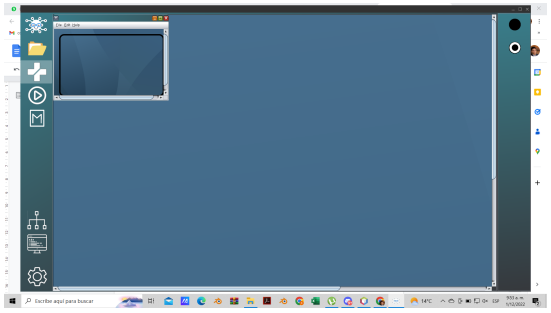
- Manual de usuario:



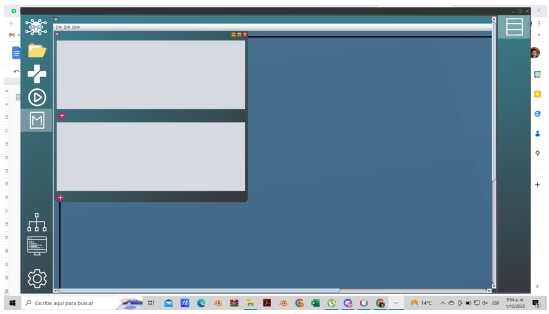
los botones de la parte inferior izquierda te permiten alternar entre el creador de diagramas y el creador de código



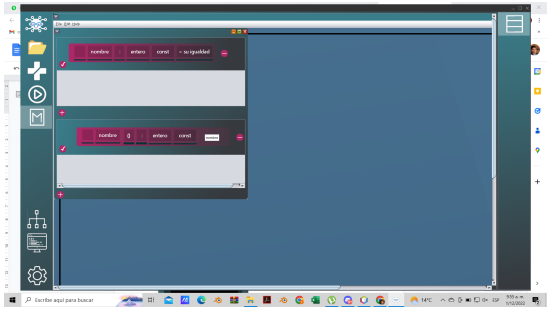
El botón señalado te permite cambiar entre uml diagrama de clases y diagrama de actividades, las dos principales de arduino a nivel principiante



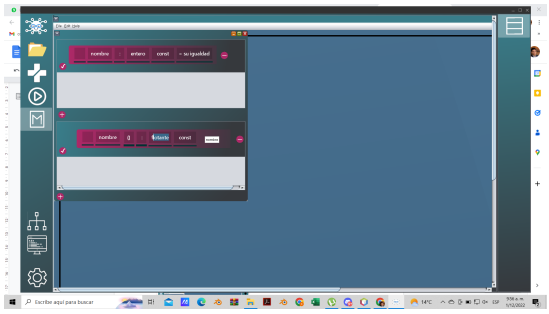
el botón más te permite abrir tantos archivos como quieras.



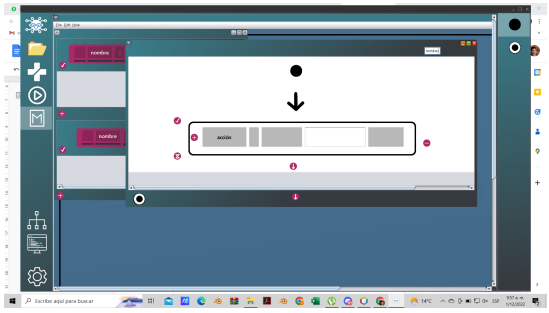
el botón de clase te generará una pestaña para crear atributos y crear métodos.



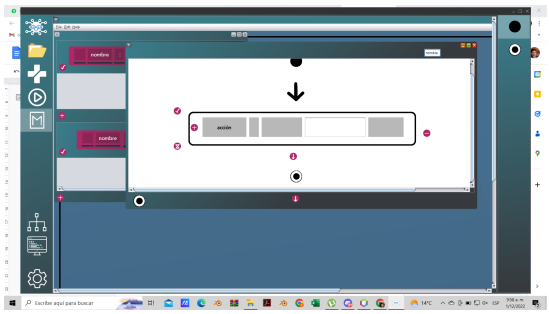
los más generan instancias que deben confirmarse presionando el chulo, se pueden quitar dinámicamente con el botón menos.



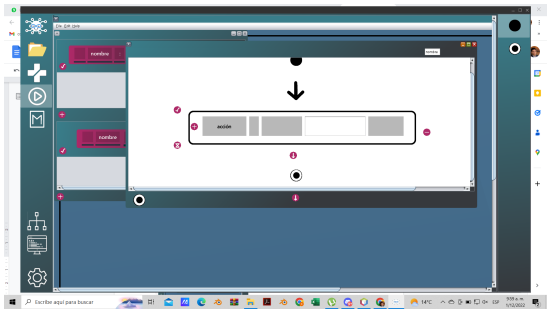
Cada espacio importante tiene autorrelleno.



lo mismo en clases pero debes undir el mas para cargar la plantilla correspondiente para la creación de la acción mientras la aprendes.



el botón final, la bola negra y blanca finaliza las acciones de ese diagrama.



el botón más crea el código.

- Conclusiones

El trabajo fue enriquecedor aunque debemos entender cuando algo está tomando mucho tiempo y más bien avanzar en otra parte del plan.

- Referencias Bibliográficas

https://docs.oracle.com/javase/7/docs/api/

https://mvnrepository.com/