Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación
Programación II

Sistema Graficación de Figuras

“Amadeus”

Manual Técnico

Arizmendi Ramírez Esiel Kevin

Coria Rios Marco Antonio

Ruiz Lozano Paulo Cesar

Otoño 2018

Contenido

Análisis

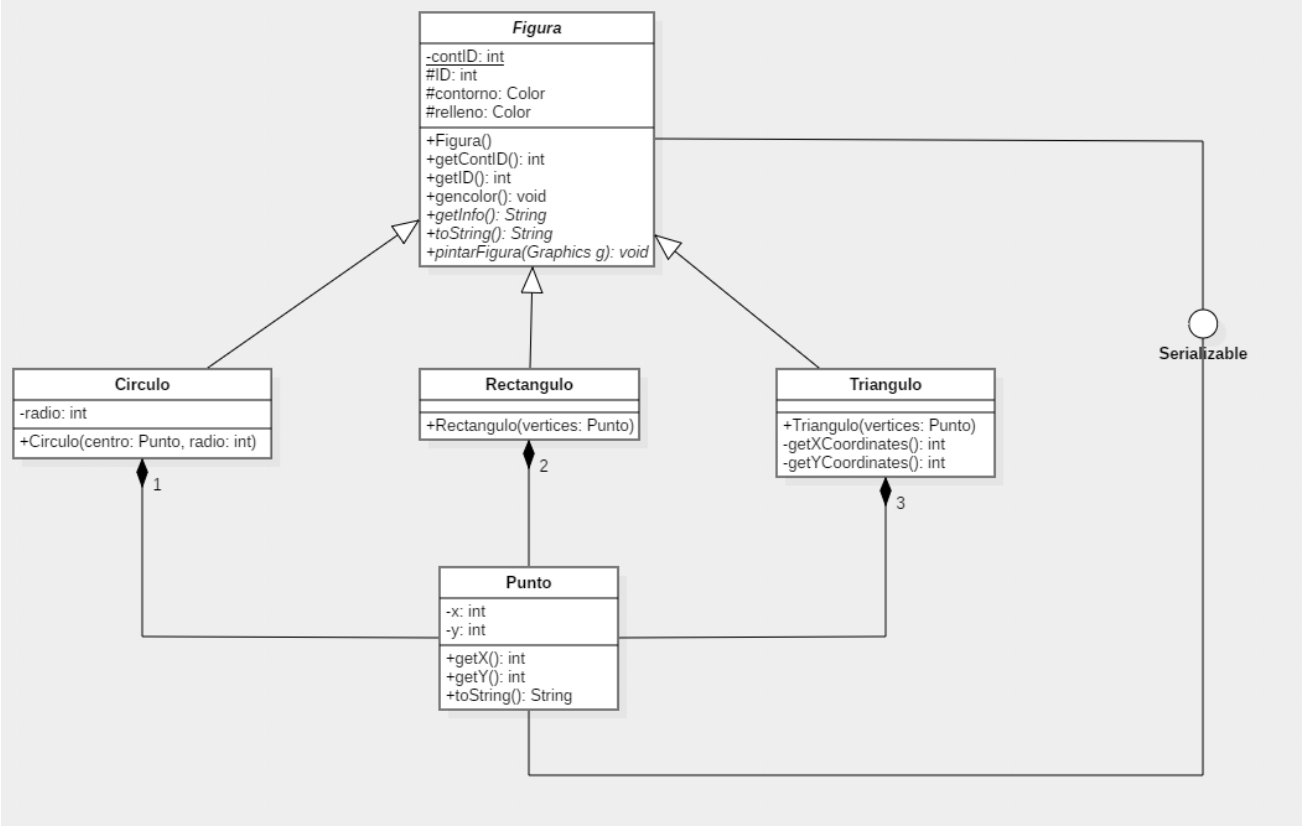
Programa

Se requiere la realización de un programa escrito con el lenguaje Java en entorno gráfico que simule la creación de tres figuras geométricas (rectángulos, círculos y triángulos) basándose en las coordenadas de los puntos y/o elementos de dichas figuras.

Para el rectángulo se requieren dos de sus puntos, el punto superior izquierdo y el punto inferior derecho, para el círculo se requieren el punto que será su centro y el tamaño del radio y para el triángulo se requieren sus tres puntos.

El programa deberá permitir guardar el trabajo realizado en archivos binarios, así como también cargarlos al programa y editarlos si se requiere.

Propuesta para su solución



(Diagrama UML de la representación para las figuras)

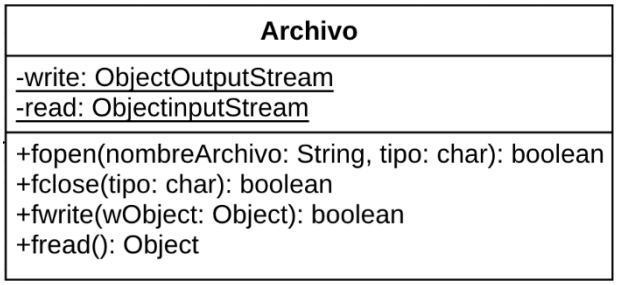
Ya que se habla de figuras geométricas que guardan comportamientos y atributos en común realizamos la abstracción de las tres figuras, creando la clase abstracta *Figura.*

Se menciona que cada figura *tiene* (esta compuesta) al menos un *punto,* y ese punto es otro objeto que tiene coordenadas tanto en el eje X como en el eje Y, entonces creamos la

clase Punto que como atributos tendrá tanto la coordenada X, como la coordenada Y, y métodos para obtenerlos. Se puede notar que las figuras tienen al menos un punto en común, pero para facilitar el manejo de cada figura los puntos se guardaran en cada una de ellas y no por herencia de la clase padre.

De la clase *Figura* heredaran la clase *Circulo*, *Rectangulo* y *Triangulo* las cuales cada una tendrá sus propios atributos. La clase *Circulo* tiene como atributos propios el radio y un *punto*. La clase *Rectangulo* tiene como atributos propios un arreglo de *puntos* llamado *vertices* el cual guardara los dos *puntos* requeridos para su creación. La clase *Triángulo* tiene como atributos propios un arreglo de *puntos* llamado *vertices* el cual guardara los tres *puntos* requeridos para su creación.

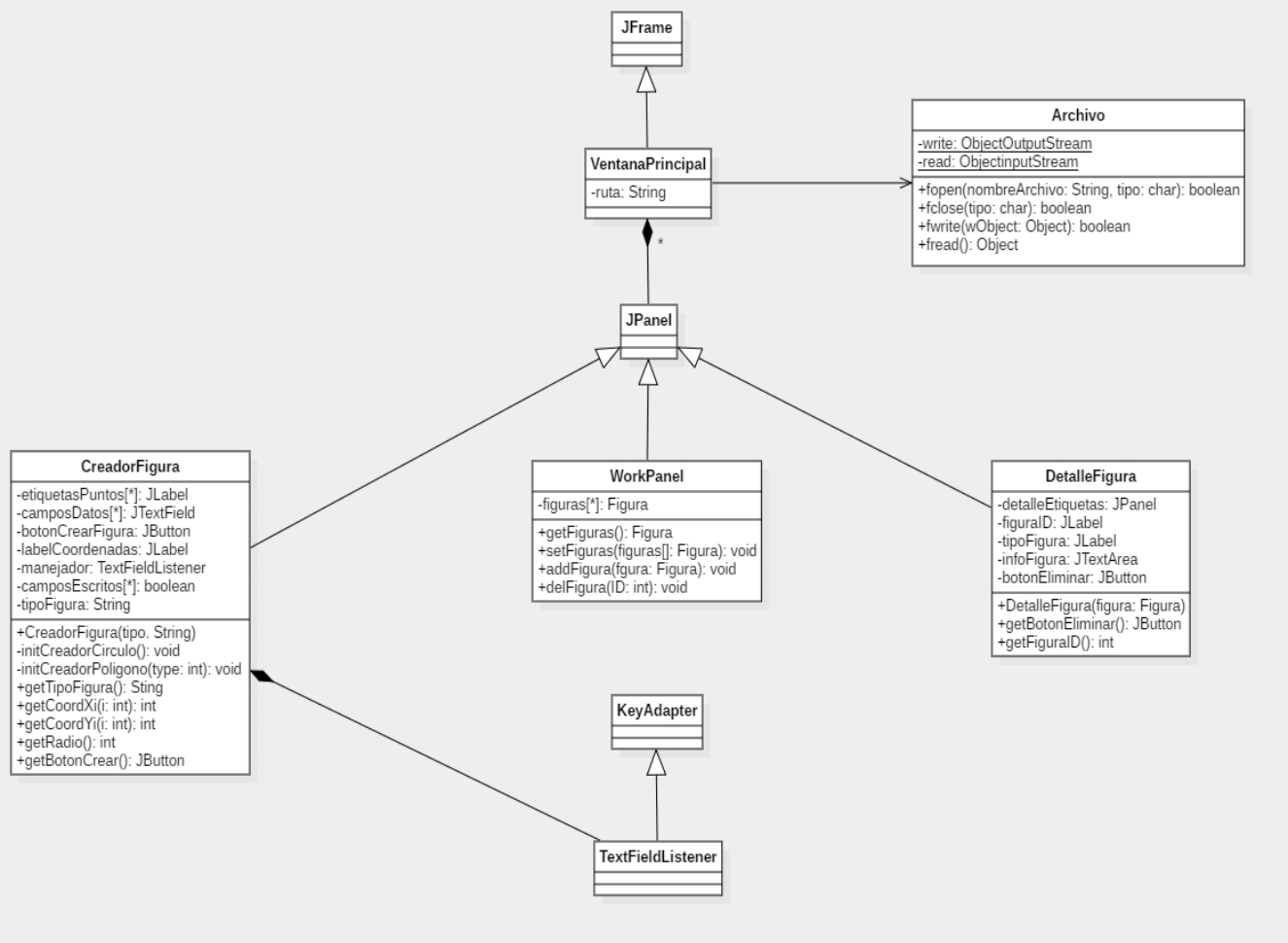
Como parte de la interfaz gráfica (GUI) cada figura tendrá la responsabilidad de pintarse en un componente gráfico, utilizando un método para generar un color aleatorio para sus bordes y el relleno de la misma.



Para el manejo de archivos binarios se requiere la implementación de la *interface Serializable* en las clases que se guardaran en los archivos. Y así la clase Archivo facilitará la manipulación de los archivos con variables estáticas privadas y métodos estáticos.

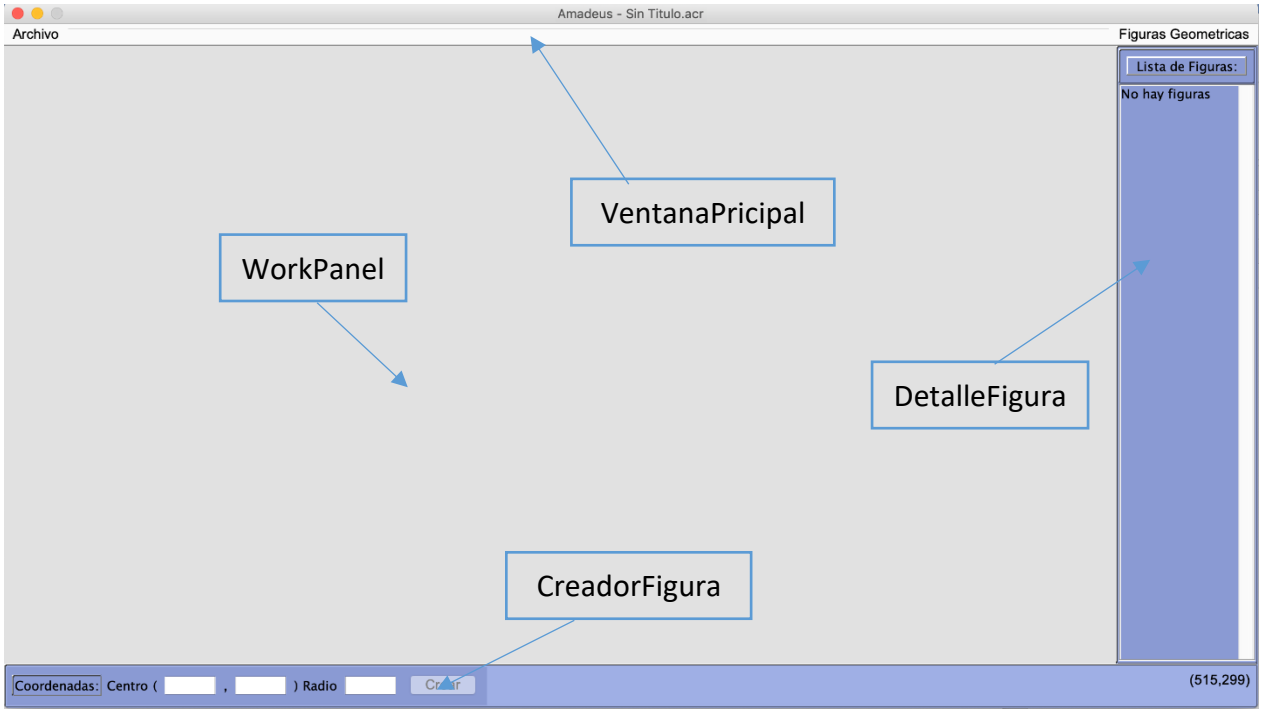
En java existen las clases *ObjectOutputStream* utilizada para escribir archivos binarios y *ObjectInputStream* utilizada para leer archivos binarios; Ya que se implementa la interface *Serializable* a las clases a guardar es de vital importancia leerlos y escribirlos de manera correcta, de lo contrario los archivos podrían ser guardados de manera errónea y no se podría recuperar su contenido.

Propuesta de Interfaz Gráfica



(Diagrama UML de la representación para la interfaz gráfica)

Para lograr una interfaz acertada y ordenada, procedimos a la especialización de componentes existentes de la librería *javax.swing*, *JFrame y JPanel*, con JFrame especializamos ununa nueva ventana, *VentanaPrincipal* que tendrá un atributo donde se guardara la ruta del archivo ejecutándose, y con el JPanel especializamos tres nuevos paneles creadorFigura, WorkPanel, DetalleFigura con el fin de tener una mayor organización en la interfaz.



(Paneles especializados de la interfaz gráfica, MacOS Mojave 10.14.1)

VentanaPrincipal:

Ventana que contiene todos los JPanel, JPanel especializados y menú de Archivo y selección de Figuras Geométricas.

WorkPanel:

Panel en donde se mostrarán las figuras creadas en colores aleatorios.

CreadorFigura:

Panel que permitirá al usuario escribir los datos requeridos para la creación de la figura seleccionada en Figuras Geométricas con la ayuda de las coordenadas ubicadas en la parte inferior derecha.

DetalleFigura:

Panel que mostrara información de las figuras creadas en forma de lista.

Nota: Los Diagramas se encuentran en formato de imagen en la carpeta “Diagramas” dentro del directorio del programa.

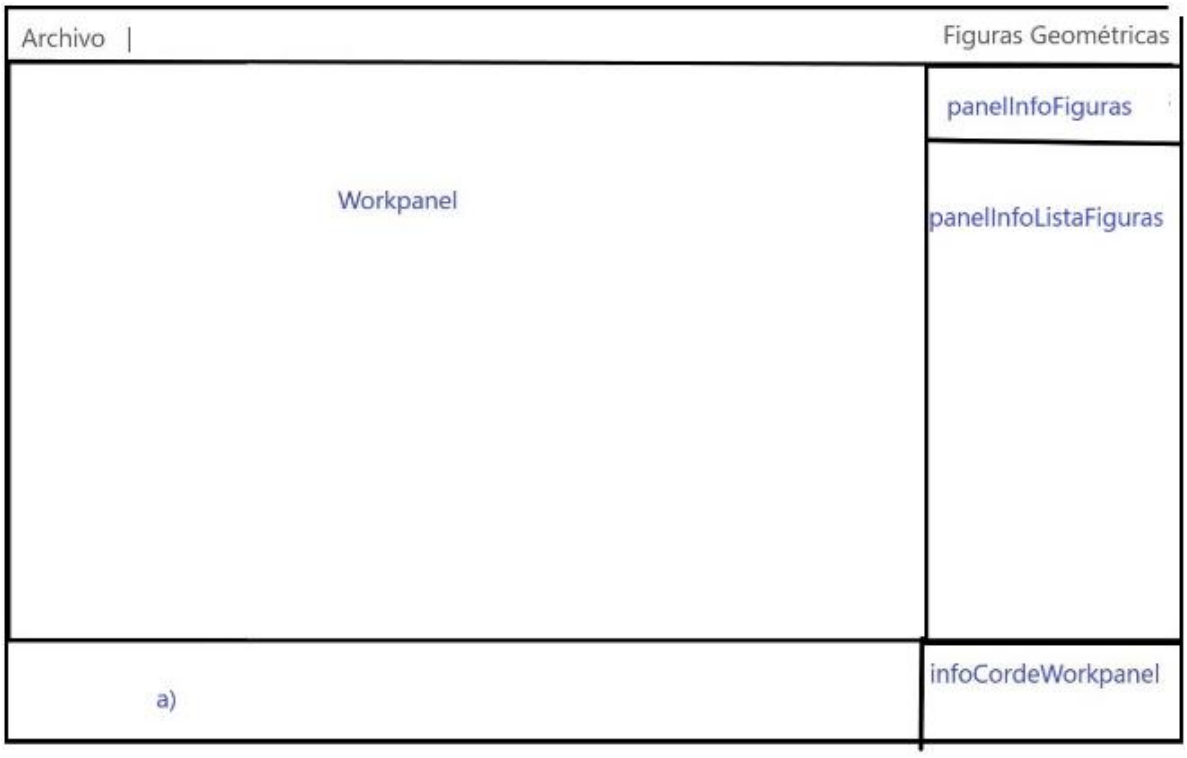
Desarrollo

El proyecto “Amadeus”, fue desarrollado sin el uso de Entornos de Desarrollo Integrados (IDE) tales como NetBeans o Eclipse; cada línea de código está escrita manualmente en editores de texto enfocados en la programación (como Atom) y compilado a través de los comandos en terminal que ofrece el Kit de Desarrollo de Java (JDK).

A pesar del no uso de IDEs, sí que se siguió un modelo de Programación de Eventos similar al usado por NetBeans, en el que se utilizan clases anónimas para definir los escuchadores de eventos, y funciones dentro de una clase principal para poder acceder a todos los elementos de un cierto conjunto (por ejemplo, los elementos de la Ventana Principal).

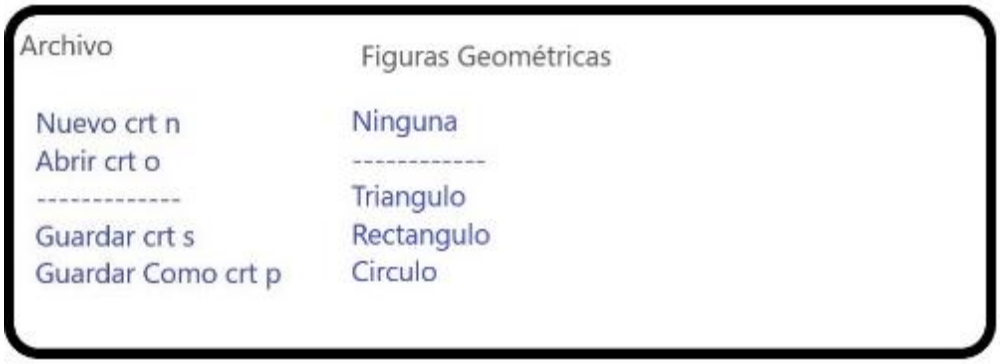
Desarrollo del entorno gráfico

Durante el proceso de desarrollo del proyecto, se debatieron diferentes ideas del como debería lucir el programa, considerando crear una aplicación entendible y poco cargada de elementos; finalmente se optó por una idea igual al siguiente diagrama:



(Diseño de la Ventana Principal)

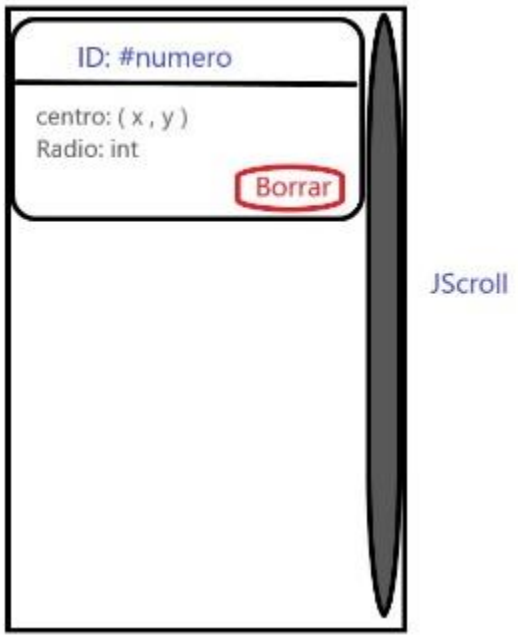
A la par, fueron pensadas las funcionalidades que tendría la aplicación, y que serían colocadas en el menú:



(Opciones de los menús planificadas para la aplicación)

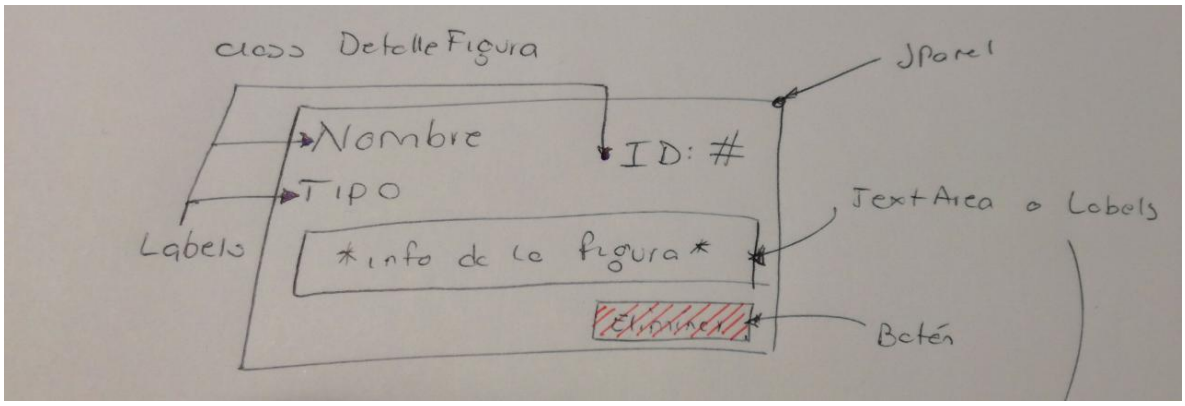
Parte de los elementos del entorno gráfico se agruparon en secciones comunes, donde se irían intercambiando sus elementos para ofrecer un entorno poco cargado visualmente, y dando solo los recursos necesarios para cada actividad:

(Secciones para ingresar los datos al momento de crear una Figura)



(Visualización de los detalles de las Figuras creadas)

Se detalló también una forma en la cual el usuario pudiera visualizar información sobre las figuras que iba creando, y además que pudieran ser eliminadas. Se consideró el caso de agregar paneles individuales con los detalles de una figura (Tipo, identificador, información concreta de las coordenadas) más su botón individual para ser eliminada:



(Diseño del panel de detalles para una Figura)

Ya con un diseño para la interfaz gráfica, se procedió a empezar con la creación del código, partiendo con la creación de las clases para la resolución del problema (Diagrama de las clases de Figuras), y desarrollando una base para el entorno gráfico a la par.

Desarrollo del Programa

Las clases que se muestran en los diagramas son:

- Archivo
- Circulo
- CreadorFigura
- DetalleFigura
- Figura
- Punto
- Rectángulo
- Triángulo
- VentanaPrincipal
- WorkPanel

Cada una con una función específica dentro del programa, el cual está descrito a continuación:

Archivo

La clase Archivo funciona para el manejo de archivos binarios. Se ha tomado la extensión “ARC” para los archivos de tipo binario que guardan la información de las Figuras Geométricas creadas en la aplicación.

La clase Archivo es una clase reutilizada de otros proyectos, que tiene un funcionamiento general para manejar flujos de datos en binario.

Recursos

- Importados:
    - java.io.FileOutputStream
    - java.io.FileInputStream
    - java.io.ObjectOutputStream
    - java.io.ObjectInputStream
    - java.lang.Exception
    - java.lang.NullPointerException
    - java.io.IOException
    - java.io.FileNotFoundException

Atributos

- write (public, de Clase, ObjectOutputStream): Flujo para salida de datos.
- read (public, de Clase, ObjectInputStream): Flujo para entrada de datos.

Métodos

V*	Función	Tipo	Definición
+	fopen()	void	Permite abrir un archivo en forma de escritura o lectura, usando sintaxis del lenguaje C: "r" para abrir un archivo en lectura, y "w" para abrir un archivo en forma de escritura.
+	fclose()	boolean	Cierra el archivo abierto
+	fwrite()	boolean	Permite la escritura sobre el archivo abierto, almacenado en la variable "write". Escribe al archivo. Retorna “true” si fue posible la escritura, o “false” de lo contrario.
+	fread()	Object	Lee un Object del archivo abierto y retorna dicho objeto.

*Visibilidad

Figura (Abstracta)

La clase Figura representa de manera general la composición y el comportamiento de una figura geométrica, que será pintada y visualizada en la aplicación.

Recursos:

- Importado:
  - java.util.Random
  - java.awt.Graphics
  - java.awt.Color
- Implementado:
  - java.io.Serializable

Atributos

- contID(privado, de Clase, entero): mantiene la cuenta de las figuras creadas.
- ID(protegido, final, entero): identificador de la figura.
- Contorno(protegido, Color): color de Contorno de cada Figura.

- Relleno(protegido, Color): color de Relleno de cada Figura.

Donde contorno y relleno son elementos de la clase Color (biblioteca java.awt), que permiten una visualización gráfica para el usuario.

Métodos

V	Función	Tipo	Definición
+	Figura()		Constructor
+	getContID()	int	Estático; retorna el contador ID
+	setContID()	void	Estático; asigna un valor al contador ID
+	getID()	int	Retorna el valor de la variable ID
-	genColor()	void	Llenena de color aleatorio las figuras geométricas
+	toString()	String	Abstracto
+	getInfo()	String	Abstracto
+	pintarFigura()	void	Abstracto

Circulo

La Clase Circulo, clase hija de la Clase Figura, es la representación en concreto de un círculo, compuesto por un punto central y un radio.

Recursos:

- Importado:
  - java.awt.Color
  - java.awt
  - java.io.Serializable

Atributos

Circulo está compuesta por 2 atributos, más los heredados por figura.

- Centro(privado, final, Punto): coordenadas del Centro
- Radio(privado, final, entero): radio del círculo.

Métodos

V	Función	Tipo	Definición
+	Circulo()		Constructor
+	toString()	String	Retorna la información en forma String
+	getInfo()	String	Retorna parte de la información en forma String
+	pintarFigura()	void	Pinta la Figura en un Panel

Rectangulo

La Clase Rectángulo, clase hija de la Clase Figura, es la representación en concreto de un Rectángulo, que se compone de dos puntos. Dichos puntos son sus esquinas Superior Izquierda e Inferior Derecha, con los que se obtienen las cuatro esquinas.

Recursos:

- Importado:
  - java.awt
- Implementado:
  - java.io.Serializable

Atributos

- Punto[] Vértices: Donde se almacenan los vértices que componen la figura

Métodos

V	Función	Tipo	Definición
+	Rectángulo()		Constructor
+	toString()	String	Retorna la información en forma String
+	getInfo()	String	Retorna parte de la información en forma String
+	pintarFigura()	void	Pinta la Figura en un Panel

Triangulo

La clase Triangulo, clase hija de la Clase Figura, es la representación en concreto de un Triángulo. Se compone de tres puntos, siendo éstos sus esquinas.

Recursos:

- Importado:
  - java.awt
- Implementado:
  - java.io.Serializable

Atributos

- Punto[] Vértices: Donde se almacenan los vértices que componen la figura

Métodos

V	Función	Tipo	Definición
+	Triángulo()		Constructor
+	toString()	String	Retorna la información en forma String
+	getInfo()	String	Retorna parte de la información en forma String
+	pintarFigura()	void	Pinta la Figura en un Panel
-	getXCoordinates()	int []	Obtiene coordenadas “x” del arreglo de puntos
-	getYCoordinates()	int []	Obtiene coordenadas “y” del arreglo de puntos

Punto

La Clase Punto, representa las coordenadas de un punto en el plano.

Recursos:

- Implementa:
  - java.io.Serializable

Atributos

- X(final, entero): Valor entero que representa la coordenada x en el plano.
- Y(final, entero): Valor entero que representa la coordenada y en el plano.

Métodos

V	Función	Tipo	Definición
+	Punto		Constructor: Recibe dos coordenadas
+	getX()	int	Retorna el valor x
+	getY()	int	Retorna el valor y
+	toString()	String	Retorna información en forma de String

VentanaPrincipal

La Clase VentanaPrincipal es la clase sobre la que se inicia el programa. Contiene al método main y todos los elementos gráficos que se muestran, además de las variables de control del programa.

Aquí es donde se definen también métodos para la implementación de las clases anónimas de los escuchadores de eventos.

Recursos:

- Importado:
  - java.util.ArrayList;
  - javax.swing..*
  - javax.swing.filechooser
  - javax.swing.border
  - java.awt.*
  - java.awt.event
  - java.io.File

Atributos

- Constantes
  - WINDOW_LENGHT(Privado): Representa el largo de la ventana.
  - WINDOW_HEIGTH(Privado): Representa el ancho de la ventana.
  - PANEL_BACKGROUND(Privado): representa el color azul claro.
  - SUBPANEL_BACKGROUND(Privado): Representa el color azul cian.
- Variables
  - detallesFiguras(Privado): Lista ligada que contiene a todas las Figuras Geométricas creadas.

- Ruta(Privado): Contiene la ruta donde se almacena el archivo actual
- Cambios(Privado): Bandera de cambios en el programa, para actualizar el estatus de guardado o no guardado
- creadorTrianguloHabilitado(Publico): Bandera que regula el cambio del panel creadorFiguras.
- creadorRectanguloHabilitado(Publico):Bandera que regula el cambio del panel creadorFiguras.
- creadorCirculoHabilitado(Publico): Bandera que regula el cambio del panel creadorFiguras.

- Componentes Gráficos
  - contenedorPrincipal(Privado): es donde se pondrán todos los demás componentes gráficos.
  - workPanel(Privado): Panel donde se pintarán las figuras Geométricas.
  - coordenadasInfo(Privado): Un etiqueta (texto sin forma) que informa las coordenadas actuales del ratón, sobre el workpanel.
  - creadorFiguras(Privado): Panel donde van a intercambiase diferentes paneles, para crear figuras, dependiendo de las acciones del usuario.
  - panelInfo(Privado): Etiqueta que informa al usuario de que el panel cambia dependiendo del clic que haga sobre el menú.
  - scrollInfoFigura(Privado): Panel Adaptable a agregar más información.
  - panelInfoListaFiguras(Privado): Panel que muestra información en forma de lista, todas las figuras creadas.
  - panelInfoFigura(Privado): Panel que tiene texto explicando a usuario diferente tipo de información.
  - coordenadasWorkPanel(Privado): Panel donde se almacenan las coordenadas dadas por la etiqueta “coordenadasInfo”.
  - panelCrearDefault(Privado): Panel por defecto en el panel “creadorFiguras”.
  - panelCrearTriangulo(Privado): Panel que permite al usuario generar triángulos sobre el workpanel.
  - panelCrearRectangulo(Privado): Panel que permite al usuario generar rectángulos sobre el workpanel.
  - panelCrearCirculo(Privado): Panel que permite al usuario generak círculos sobre el workpanel.

- Menus
    - menuPrincipal(Privado): Barra principal donde se trabajaran diferentes menús para el manejo del archivo actual.
    - menuArchivo(Privado): Menú archivo, que almacena opciones para el manejo del archivo actual, tales como: guardar, abrir, nuevo.
    - menuArchivoNuevo(Privado): Opción del panel menú archivo para generar un nuevo archivo.
    - menuArchivoAbrir(Privado): Opción del panel menú archivo para abrir un archivo con información previamente guardada.
    - menuArchivoGuardar(Privado): Opción del panel menú archivo para guardar el estado actual del archivo que se trabaja.
    - menuArchivoGuardarComo(Privado): Opción del panel de menú archivo, para guardar con un nombre en específico.
    - menuFigurasGeometricas(Privado): Menú de figuras geométricas, con el cual se pueden crear en el workpanel.
    - menuFigurasDefault(Privado): Opción, que regresa a estado de inicio del programa sin eliminar las figuras ya creadas en el workpanel.
    - menuFigurasTriangulo(Privado): Opción del menú Figuras Geométricas, que permite cambiar el panel creadorFiguras, para que el usuario pueda crear triángulos
    - menuFigurasRectangulo(Privado):
    - menuFigurasCirculo(Privado):

Métodos

V	Función	Tipo	Definición
+	VentanaPrincipal()		Constructor: carga los componentes gráficos y llama a initComponents e initMenus.
-	initComponents()	void	Inicia los paneles y los componentes de estos.
-	initMenus()	void	Inicia el menú principal del programa. Requiere de dos argumentos, las fuentes de los menus,

+	cambiarCreadorFiguras()	void	Función encargada de controlar el cambio de los paneles inferiores, que pueden crear figuras geométricas Requiere de un argumento JPanel,
+	actualizarPanelInfo()	void	Actualiza la información que contiene el panel lateral izquierdo, donde se muestra la información necesaria del programa,
+	mostrarAdvertencia()	void	Función que permite regular las acciones del usuario cuando este intenta crear un nuevo archivo sin haber guardado. Requiere un argumento String para las diferentes modalidades de advertencia.
+	mostrarSeleccion()	int	Requiere un argumento String para el control de la función
+	guardarComoArchivo()	void	Función que Permite guardar el archivo bajo un nombre y una dirección.
+	reiniciarFrame()	void	Función que permite limpiar todo el entorno gráfico.
+	abrirArchivo()	boolean	Función que permite cargar un archivo desde una ruta.
-	salir()	void	Función que verifica que la salida del usuario no tenga pérdida de datos.
Funciones de Eventos			
+	menuArchivoNuevoMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Nuevo”.
+	menuArchivoAbrirMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Abrir”.
+	menuArchivoGuardarMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Guardar”.

+	menuArchivoGuardarComoMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Guardar Como”.
+	menuFigurasDefaultMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Ninguna”.
+	menuFigurasCirculoMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Circulo”.
+	menuFigurasTrianguloMouseClicked()	Void	Evento que escucha cuando se da clic sobre el menu “Triángulo”.
+	menuFigurasRectanguloMouseClicked()	void	Evento que escucha cuando se da clic sobre el menu “Rectángulo”.
+	workPanelMouseMoved()	void	Evento que identifica la posición del mouse sobre el panel de trabajo.
+	botonCrearMouseClicked()	void	Evento que controla cuando se puede crear una figura geométrica.
+	crearCirculo()	Circulo	Evento que regula la creación y dibujo de un círculo.
+	crearRectangulo()	Rectángulo	Evento que regula la creación y dibujo de un rectángulo.
+	crearTriangulo()	Triangulo	Evento que regula la creación y dibujo de un triángulo.
+	botonEliminarMouseClicked()	void	Evento que controla cuando se puede eliminar una figura geométrica
+	main	void	Programa Principal

WorkPanel

La Clase WorkPanel, es la sección de trabajo para pintar todas las Figuras Geométricas que son creadas. Esta clase hereda de “JPanel” de la librería “swing”.

Recursos:

- Importado:
  - javax.swing.*
  - java.awt.*
  - java.awt.event

- java.util.ArrayList

Atributos

- figuras(privada): Arreglo dinámico (ArrayList) de todas las figuras que se crean en el programa.

Métodos

V	Función	Tipo	Definición
+	WorkPanel		Constructor. Inicial los componentes de un JPanel.
+	paint()	void	Permite pintar sobre el panel diferente figuras. (El método es reescrito de JPanel)
+	getFiguras()	ArrayList<Figura>	Obtiene la lista de todas las figuras geométricas creadas.
+	setFiguras	void	Esta función permite establecer una lista de figuras geométricas.
+	addFigura()	void	Añade un nuevo elemento a la lista de figuras geométricas y llama a “paint” para imprimirla.
+	delFigura()	void	Dado un ID como parámetro, permite eliminar la figura geométrica asociada a ese ID.

CreadorFigura

Clase Creador de Figuras. Crea los paneles para recibir información de las figuras e indicar al programa principal los datos para ser creadas.

Recursos:

- Importado:
  - javax.swing
  - java.util.ArrayList
  - java.awt
  - java.awt.event
  - javax.swing.border

Atributos

- etiquetasPuntos(privada): Una lista de todas las tiquetas para detonar las coordenadas a introducir.

- camposDatos(privada): Una lista de todas las areas de texto para las coordenadas y el radio
- botonCrearFigura(privada): Boton Creador de figuras geométricas
- private final JLabel labelCoordenadas = new JLabel("Coordenadas:");
- manejador(Privada): La variable manejador es una variable de tipo TextFieldListener, una clase privada creada dentro de la clase creador de figuras.
    - **Clase TextFieldListener**
        - Esta clase maneja todos los eventos relacionados al manejo del las áreas de texto donde se pondrán las coordenadas.
        - Esta clase hereda de “KeyAdapter” de la librería “awt.event”.

- private boolean[] camposEscritos(privada): Un arreglo de banderas que permite regular información de todas las áreas de texto; cuando fueron modificadas, si contienen caracteres.
- tipoFigura(privada): Es un String que contiene el tipo de figura geométrica con el cual se trabajará.
- Columns(privada, estatica, final): una constante de valor entero 4 que contiene la cantidad de columnas que se van a manejar en el panel.

Métodos

V	Función	Tipo	Definición
+	CreadorFigura()		Constructor: Recibe como parámetro de creación un String donde se establece el tipo de panel que se va a crear.
-	initCreadorCirculo()	void	Inicia los componentes gráficos del panel.
-	initCreadorPoligono()	void	Inicia los componentes gráficos para la entrada de datos de creación de figuras geométricas.
+	getTipoFigura()	String	Retorna el tipo de figura geométrica del cual fue creado el panel.
+	getCoordXi(int i)	int	Retorna lo escrito dentro de área de texto de la coordenada “x”. Puede retornar una exepcion de tipo ArrayIndexOutOfBoundsException
+	getCoordYi(int i)	int	Retorna lo escrito dentro de área de texto de la coordenada “y”. Puede retornar una exepcion de tipo ArrayIndexOutOfBoundsException

+	getRadio()	int	Retorna lo escrito dentro de área de texto del radio. Puede retornar una exepcion de tipo ArrayIndexOutOfBoundsException
+	getBotonCrear()	JButton	Retorna toda la información del botón crear figura.
Funciones de la clase TextFieldListener			
+	keyTyped()	void	Función que regula todas las entrada de texto generadas a partir de una tecla, compara si el carácter escrito es entero, limita la cantidad de caracteres a 4, habilita o deshabilita el botón crear figura.

DetalleFigura

La Clase DetalleFigura es heredada de JPanel, y se crea para colocar la información sobre una Figura en el sistema. Contiene la información de una Figura determinada, sin embargo, no contiene la referencia a dicha Figura.

Esta clase recibe de manera genera una instancia de tipo Figura, por lo que no se requirió hacer tres clases más para cada tipo de especialización de ella.

Recursos:

- Importado:
  - javax.swing
  - javax.swing.border
  - java.awt

Atributos

Todos los atributos de esta clase son privados y finales(No se pueden modificar).

- detalleEtiquetas: Panel donde se almacenan todos los demás atributos de la clase.
- figuraID: Etiqueta donde se escribirá el ID de cada figura geométrica cuando sea creada.
- tipoFigura: Etiqueta donde se dea información al usuario a cerca del tipo de figura que se creó.
- infoFigura: Area de texto donde se muestra toda la información restante de la figura geométrica: Cuantos vertices tiene, la información de dichos vertices; en el caso del circulo: el radio del circulo.
- botonEliminar: Este botón lo tienen todas las figuras geométricas creadas. Permite eliminar dicha figura.

Métodos

V	Función	Tipo	Definición
+	DetalleFigura		Constructor: Recibe una variable de tipo Figura, que se va a analizar toda la información y se creará su ficha de información correspondiente.
+	getBotonEliminar()	JButton	Retorna la información del botón de eliminar.
+	getFiguraID()	int	Retorna el ID de la figura geométrica

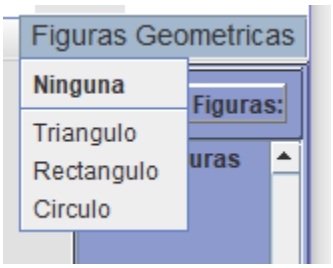
Conclusiones

Complicaciones del Desarrollo

Durante el desarrollo del proyecto se presentaron algunas complicaciones:

La descripción del problema especificaba el uso exclusivo de ventanas de diálogo para despliegue de mensajes de advertencia o error, excluyendo la lectura de datos por teclado. Por tanto, la lectura de datos para las tres figuras se debía realizar en la misma ventana, donde éstas también se mostrarían graficadas.

Para no saturar la ventana, se creó en la barra de menús, un menú específico para crear una figura, desplegando las tres opciones más la opción por defecto. Cada opción intercambia los paneles de la parte inferior, mostrando únicamente los campos receptores de datos para una sola figura.



(Menú de selección para la creación de Figuras)

(Panel para la creación de un Triángulo)

Debido a la creación de figuras geométricas por medio de coordenadas escritas por el usuario, el programa otorgaba mucha libertar para ingresar datos erróneos y provocar fallos en el sistema: letras y símbolos, coordenadas negativas o fuera del panel de trabajo, campos vacíos. El programa sólo debía recibir números y además, enteros, para tener un funcionamiento correcto.

Aunque para solucionar este problema se podría haber mostrado ventanas de diálogo con las descripciones de los errores, la solución abordada resulta más eficiente: restringir estas libertades al usuario. Así, se preparó al sistema para permitir únicamente la entrada de caracteres numéricos, y crear la figura en el momento en que todos los campos tengan algún dato escrito.

Algunos problemas más específicos con respecto al ingreso de datos fueron manejadas con cuadros de diálogo, pero su uso se redujo considerablemente.

La aplicación final resulta en un Sistema para Graficación de Figuras Geométricas, en concreto, círculos, triángulos y rectángulos, por medio de introducción de coordenadas, y, en el caso del círculo, la medida del radio.

Alcances y Limitaciones

-De las funciones de la aplicación:

El programa tiene la limitación de no poder seleccionar los colores de la figura (contorno y relleno) de forma manual, puesto que no se presentaba como una característica fundamental para la solución del problema, y su implementación habría aumentado el tiempo de desarrollo. Esta característica puede ser implementada en futuras versiones.

Debido a los recursos de Java utilizados para crear y dibujar las figuras sobre el espacio de trabajo, no existe ninguna posibilidad de “selección” de una figura ya dibujada. Las figuras que se dibujan no conforman un elemento gráfico como lo sería un botón, sino que “se pintan” sobre un panel. Por esta razón, implementar una herramienta de selección significaba más tiempo de desarrollo, añadiendo el hecho de que dicha selección se requiere de distinta forma para cada tipo de figura.

-De la portabilidad de la aplicación:

La aplicación, al estar desarrollado en el lenguaje Java, requerirá de la Máquina Virtual de Java para su ejecución. Sin embargo, se presentan problemas de compatibilidad con los elementos gráficos en plataformas distintas a Windows. Dichas incompatibilidades solo afectan a la visualización de la aplicación (componentes como botones, menús, etc.), pero no afectan al comportamiento del programa, ni generan errores o cierres abruptos.

Para guardar la información del trabajo hecho con la aplicación se han utilizado archivos binarios, con una extensión personalizada “.acr”. De esta forma, sólo la aplicación será capaz de leer este tipo de archivos, que no serán compatibles con ninguna otra. Sin embargo, no se ha determinado a profundidad si esta extensión no está siendo utilizada por otra aplicación en el mercado.