



TECNOLOGICO NACIONAL DE MEXICO  
**Instituto Tecnológico de la Laguna**  
Ingeniería en Sistemas Computacionales  
**TOPICOS AVANZADOS DE PROGRAMACION**

PERIODO: Ene - Jun / 2020

GRUPO: "B" 17 – 18 Hrs

## APUNTES DEL CURSO

ALUMNO:

18131209 José Misael Adame Sandoval

PROFESOR:

Ing. Luis Fernando Gil Vázquez

Torreón, Coah. a 12 de Junio de 2020



# INSTITUTO TECNOLOGICO DE LA LAGUNA

## INGENIERIA EN SISTEMAS COMPUACIONALES

### Curso: Topicos Avanzados de Programación

Profesor: Ing. Luis Fernando Gil Vázquez ( [fgil@universoft.com.mx](mailto:fgil@universoft.com.mx) )

**Competencia a desarrollar:** Desarrollar soluciones de software utilizando programacion concurrente, programacion de eventos, que soporte interfaz grafica.

**Competencias previas:** Diseñar e implementar objetos de programación que permitan resolver situaciones reales y de ingeniería.

#### T E M A S

##### I. Interfaz Grafica de Usuario.

1.1 Creacion de interfaz grafica para usuarios, 1.2 Computacion gráfica

##### II. Creación de componentes y librerías.

2.1 Deficion conceptual de componentes, paquetes/librerías, 2.2 Uso de componentes ( visuales y no visuales ) proporcionados por el lenguaje, 2.3 Uso de librerías proporcionadas por el lenguaje, 2.4 Creacion de componentes ( visuales y no visuales ) definidos por el usuario, 2.5 Tipos de eventos, 2.6 Generacion y propagacion de eventos, 2.7 Creacion de eventos.

##### III. Acceso a bases de datos.

3.1 Introduccion, 3.2 Conexion a orígenes de datos, 3.3 Manipulacion de datos, 3.4 Visualizacion de datos

##### IV. Programacion concurrente multihilo.

4.1 Concepto de hilo, 4.2 Comparacion de un programa de flujo unico contra uno de flujo multiple, 4.3 Creacion y control de hilos, 4.4 Sincronizacion de hilos.

##### V. Proyecto final.

##### Criterios de Evaluación:

Criterio	Porcentaje
Unidad 1 al 4 : Examen Teorico/Practico	40%
Unidad 1 al 4 : Practicas	40%
Unidad 1 al 4 : Participacion y asistencia	20%
Unidad 5 : Proyecto final (obligatorio)	100%
Rescate (opcional): Libreta de apuntes completos digitalizados en PDF	1 Unidad (excepto la U5)

##### Bibliografía:

1. Bell, D. & Parr, M. (2011). **Java para estudiantes.** Pearson Educación. 2. Deitel & Deitel. (2012). **Cómo programar en Java.** 9<sup>a</sup> Ed., Pearson. 3. Arnold, K. & Gosling, J. (1997). **El lenguaje de programación Java.** Serie Java. Addison-Wesley / Domo. 4. Davis, S. R. (1997). **Aprenda Java ya.** Microsoft Press. 5. Wang, P. S. (2000). **Java.** Ed. Thomson. 6. Gil, F. (2008). **Antologías: Fundamentos del lenguaje Java.** Torreón, Coah. México : Tecnológico de La Laguna. 7. Barnes, D. & Kolling, M. (2013). **Programación Orientada a Objetos con Java usando BlueJ.** 5<sup>a</sup> Ed. Pearson.

**Plataforma moodle:** <https://catedig.itlalaguna.edu.mx> Contraseña: m3canso6anso=)

Al registrarse como usuario de moodle llenar el campo "Nombre" con su numero de control, el campo "Apellido" con su nombre completo empezando por los apellidos.

**Lista de útiles:** 1. Libreta de apuntes de la materia, 2. USB exclusiva para la materia. 3. JDK 8 / Netbeans 8.0+

##### Reglas a seguir:

0. Las damas entran primero. 1. Respeto a los compañeros y al profesor, **no decir malas palabras.** 2. **No se permite contestar llamadas o chatear durante clase.** 3. Celulares en modo vibración, 4. Mantener el orden del aula o sala de laboratorio: **no dejar basura, dejar butacas y sillas acomodadas,** 5. Se permite el uso de laptop en clase para asuntos relacionados con la materia, 6. Asistir regularmente al menos 80% de las clases para derecho a examen, 7. Envio de tareas y practicas se hará en formato electrónico únicamente a través de la plataforma moodle (no enviar tareas al correo del profesor ), 8. Quebrantar las reglas 1 y 2 conlleva marcar como falta esa clase más una **suspensión de 3 clases**. 9. Esperar al profesor hasta 25 min., 10. NO esperar al profe en el pasillo de aulas, 11. Inges que no estén atentos a la clase o haciendo otras cosas se le marcará falta sin avisarle. 12. Trabajos copiados repreuba el tema completo. 13. Proyecto final copiado repreuba el curso. 14. No es necesario pedir permiso para salir al baño.

::

*José Misael Adame Sandoval*

*J. Misael Adame*

27/Ene/2020

## Tópicos Avanzados de Programación

Profesor: Ing. Fernando Gil

¿Qué les trae por acá?

Compromiso / Responsabilidad

¿A qué vengo al Tec?

Aprender

### Apuntes

- Fecha
- Título / subtítulo tema
- Letra legible
- Plumas color

Nombre: 85360673

Apellido: Gil Vázquez Luis Fernando

29 - Enero - 2020

- Entrar a moodle y bajar el examen de diagnóstico
- Ponerle nombre y # control
- ~ Resolver examen
- Subirlo a moodle
  - Archivo RAR/ZIP llamado
  - contenido

ExDiag-xxxxx.rar  
# control

1. Word del examen
2. Código del programa

Examen Diagnóstico Tópicos
+ main (args: String[]) {static} : void
<p>ArregloTec</p> <pre>+array : int[] +ArregloTec (array: int[]) +getAscendente (array: int[]) : void :</pre>

04 - Feb - 2020 Unidad 1 Interfaz gráfica de usuario

Desarrollar una clase llamada Matemática que proporcione métodos estáticos para calcular el factorial de un número, las permutaciones de un número "n" de elementos tomando "x" de ellos y las combinaciones de "n" tomando "x".

Para probar la clase anterior desarrollar una aplicación Java con interfaz (GUI) donde se permita capturar los valores de las fórmulas y mostrar el resultado correspondiente.

### Analisis

- Factorial  $n! = 1 \times 2 \times 3 \times \dots \times n$
- Permutaciones  $P(n, x) = \frac{n!}{(n-x)!}$
- Combinaciones  $C(n, x) = \frac{n!}{x! (n-x)!}$   
 $= \frac{P(n, x)}{x!}$

$n$  y  $x$  son enteros

Factorial  $P(n, x)$  y  $C(n, x)$   
devuelven enteros

Tipo de dato básico en Java

int, double, float, char, boolean, long, short, byte entero  
real

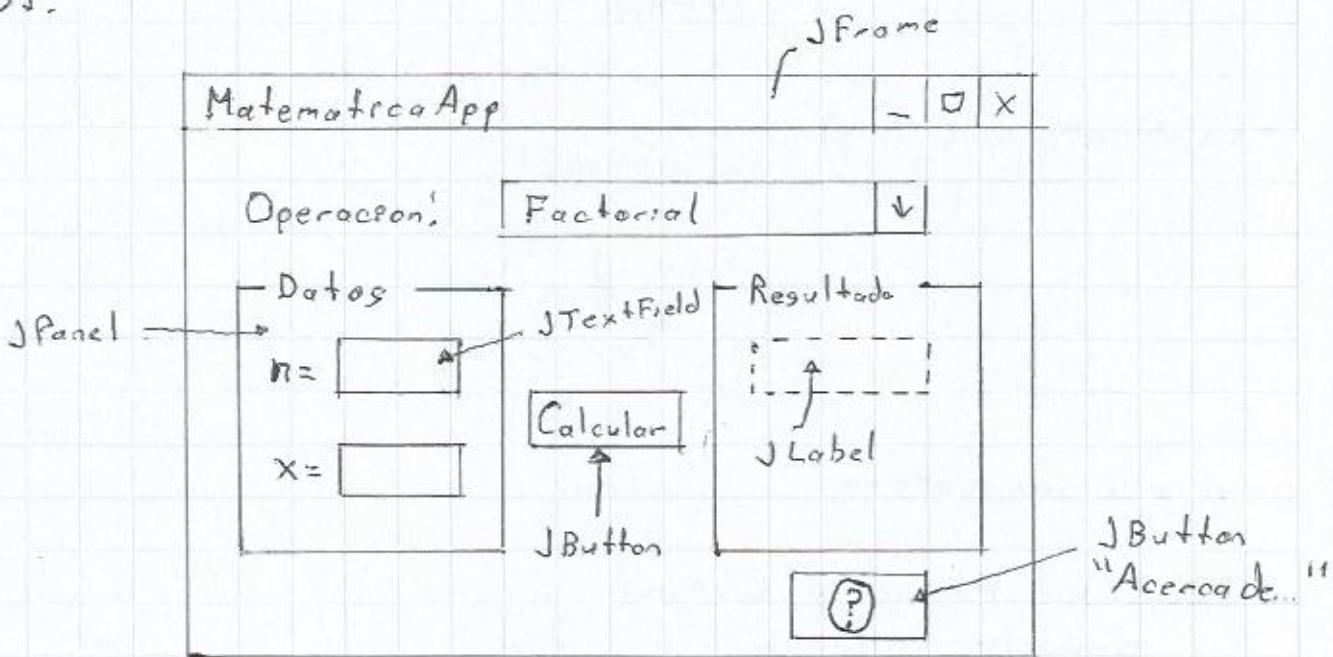
Diseno

Matemática

```
public class Matematica {
    public static long factorial (int n) { }
    public static int permutaciones (int n, int x) { }
    public static int combinaciones (int n, int x) { }
}
```

minuscula

GUI:



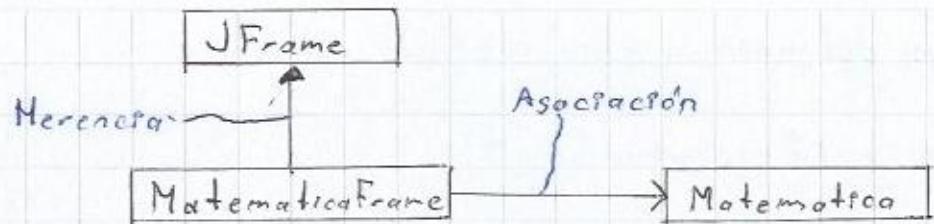


Fig. Diagrama general de clases

06- Febrero - 2020

### Validaciones:

- Factorial ( $n$ )       $0 \leq n \leq 25$  ;  $n \neq ""$
- permutaciones ( $n, x$ )  
- combinaciones ( $n, x$ )       $\left. \begin{matrix} n \geq x \\ n \neq x \end{matrix} \right\} ; n \neq ""$
- Cuadro de mensaje

`JOptionPane.showMessageDialog()`

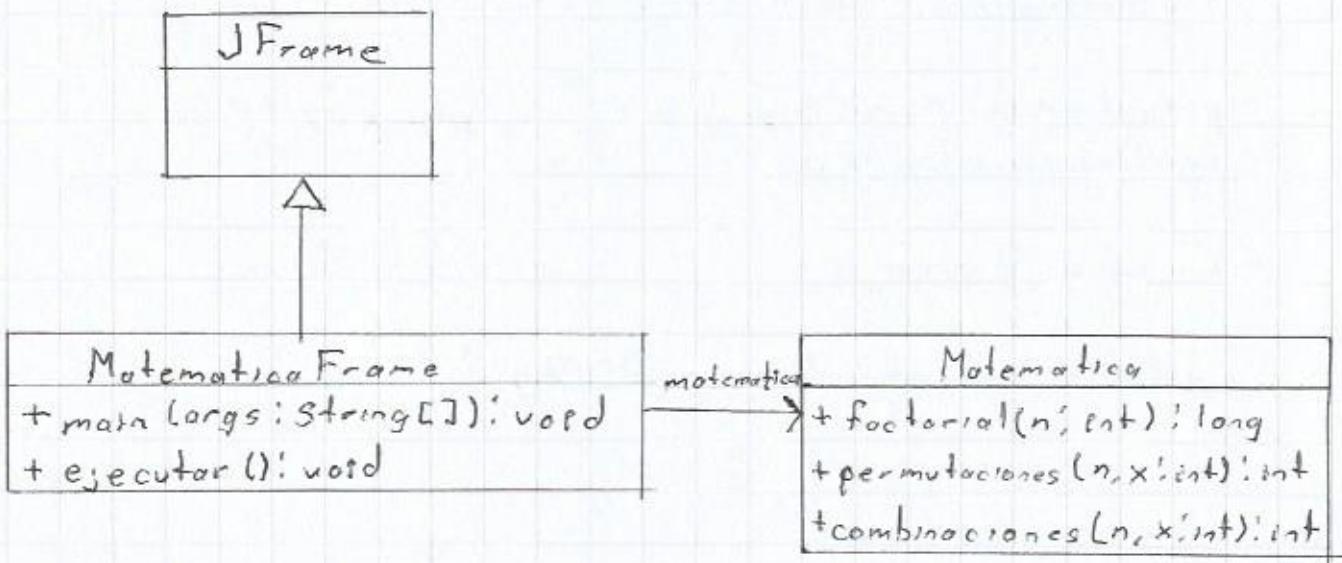
Acerca de	
Logo del Tec	Tecnológico Nacional de México Instituto Tecnológico de la Laguna
Matemática App v1.0	
Autor: Fulano de Tal xxxx	
<input type="button" value="Aceptar"/>	

07-Febrero-2020

Pasar el código del main a otro método estático

```
public static void ejecutar() {  
    ...  
}
```

```
public static void main (String args []) {  
    ejecutar();  
}
```



10-Febrero-2020

Capítulo 7-5

## JUnit

Framework para entorno de desarrollo  
Java para automatizar pruebas de unidad

- Resultado esperado } Caso de
- Resultado real obtenido } prueba

- Métodos assert xxx()

assertEquals ( String leyenda,  
                  resultado-esperado,  
                  resultado-obtenido  
        )

12-Febrero-2020

## Conversion de Unidades

Desarrollar una clase Java llamada Conversiones con métodos estáticos que permitan la conversión entre diferentes sistemas de unidades.

- De  $^{\circ}\text{C}$  a  $^{\circ}\text{F}$
- De  $^{\circ}\text{F}$  a  $^{\circ}\text{C}$
- De yardas a metros
- De metros a yardas
- De libras a kilogramos
- De kilogramos a libras
- De decimal a binario
- De binario a decimal

La clase debe declarar constantes con el factor de conversión (equivalencia) correspondiente para yardas-metros y libras-kilos.

## Análisis

→ Fórmulas

$$^{\circ}\text{F} = \frac{9}{5} ^{\circ}\text{C} + 32$$

$$^{\circ}\text{C} = \frac{5}{9} (^{\circ}\text{F} - 32)$$

⇒ Equivalencia

$$1 \text{ yarda} = 0.9144 \text{ m}$$

$$1 \text{ lb} = 0.454 \text{ kg}$$

⇒ decimal a binario

Por divisiones sucesivas

$$14 \rightarrow "1110"$$

⇒ Binario a decimal

Ej.

$$"1110" \rightarrow ?$$

$$0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 =$$

$$0 + 2 + 4 + 8 = 14$$

Método de String: charAt(int pos)

#### Conversiones

+YARDA_METRO=0.9144 {static, final}	: float
+LIBRA_KG=0.454 {static, final}	: float

17-Febrero-2020 Composición de Objetos

Cuando un objeto se compone de instancias de otros objetos

\* Ejemplo: Coche

se compone:

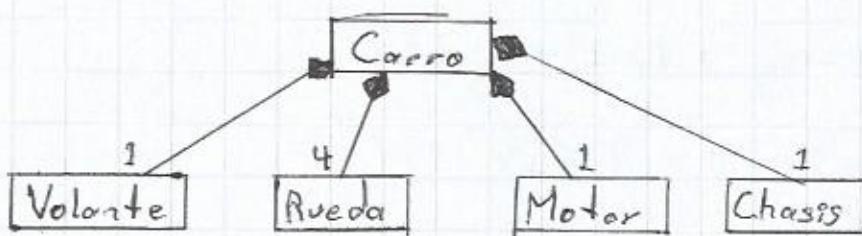
1 volante

4 ruedas

1 motor

1 chasis

◆ Composición

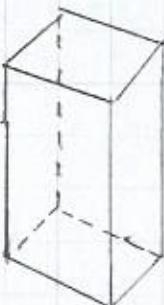


## PRISMAS

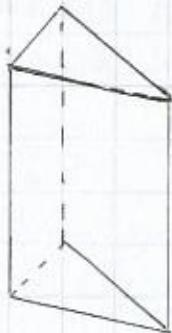
(regulares)



Cilindro



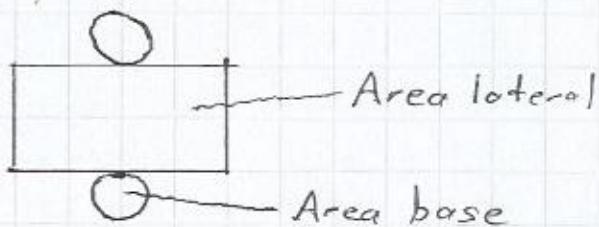
Rectangular



Triangular

Se requiere conocer el área de la base del prisma, área lateral, área total y volumen.

Círculo



$$\text{área total} = \text{área lateral} + \text{área base inferior} + \text{área base superior}$$

$$\text{volumen} = \text{área base} * \text{altura}$$

Composición

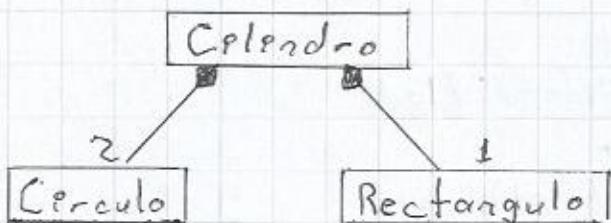


Fig. Diagrama de clases

Parámetros o valores que definen a un cilindro

- radio de la base
- altura

→ Diseño de la clase Cilindro

### Cilindro

- radio: float
- altura: float
- baseInf: Circulo
- baseSup: Circulo
- cuerpo: Rectangulo

// Constructores

+ Cilindro()

+ Cilindro(radio: float, altura: float)

// Métodos - Operaciones

+ areaBase(): float

+ areaLateral(): float

+ areaTotal(): float

+ volumen(): float

+ toString(): String

// getter / setter

+ getRadio(): float

+ setRadio(radio: float): void

+ getAltura(): float

+ setAltura(altura: float): void

U1PrimerasApp

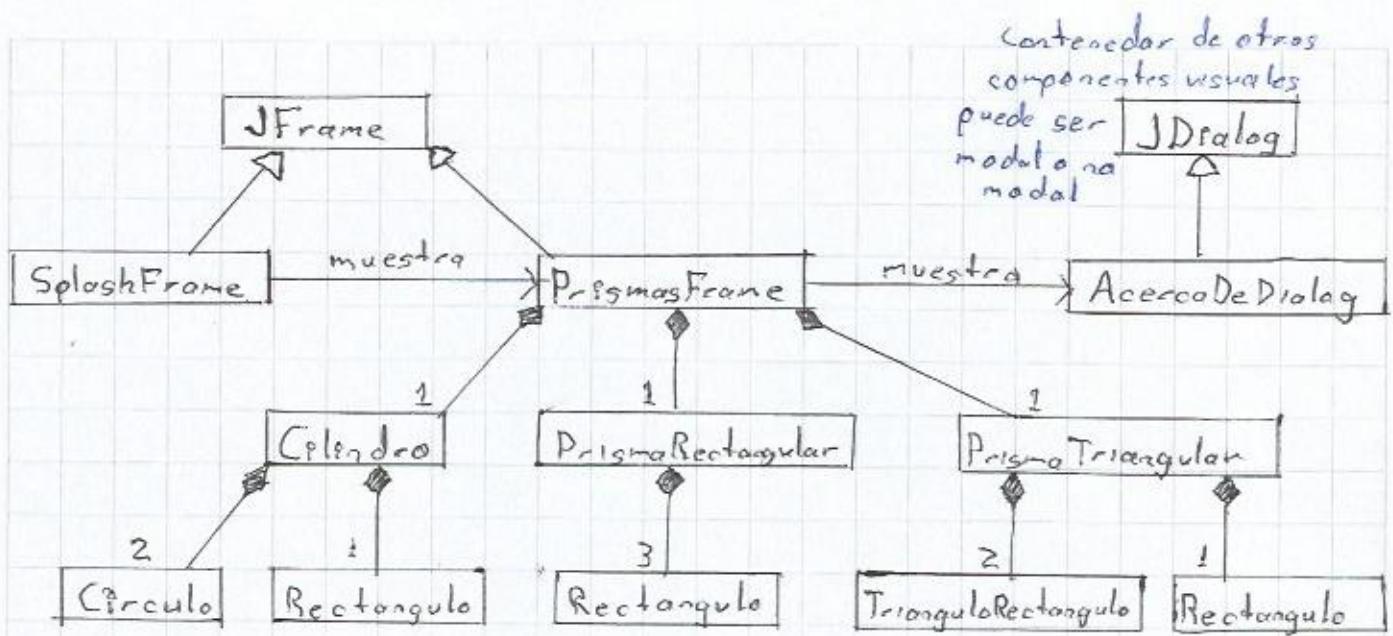
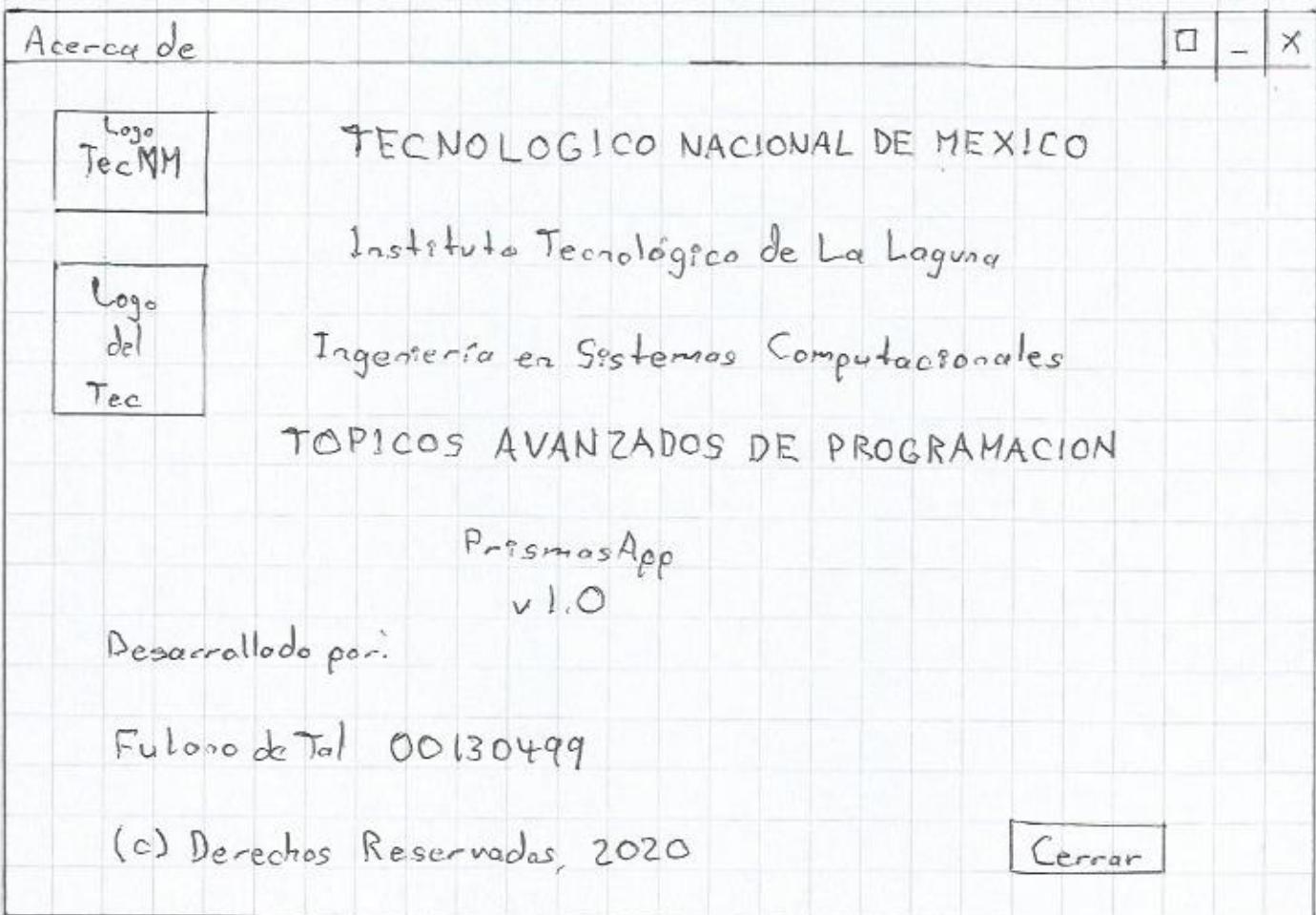


Fig. Diagrama de clases de la aplicación PrismasApp

Acerca De Dialog



JDialog (Frame parent, boolean modal)  
true

## 24-Febrero-2020 EDITOR DE TEXTO BASICO

### Características

- Permite crear un archivo nuevo
- Abrir una existente
- Imprimir el archivo
- Guardar cambios
- Copiar, cortar, pegar
- Personalizar la fuente y tamaño del texto
- Cambiar el color de la fuente y del fondo del editor
- Incluye un corrector básico.

### Análisis

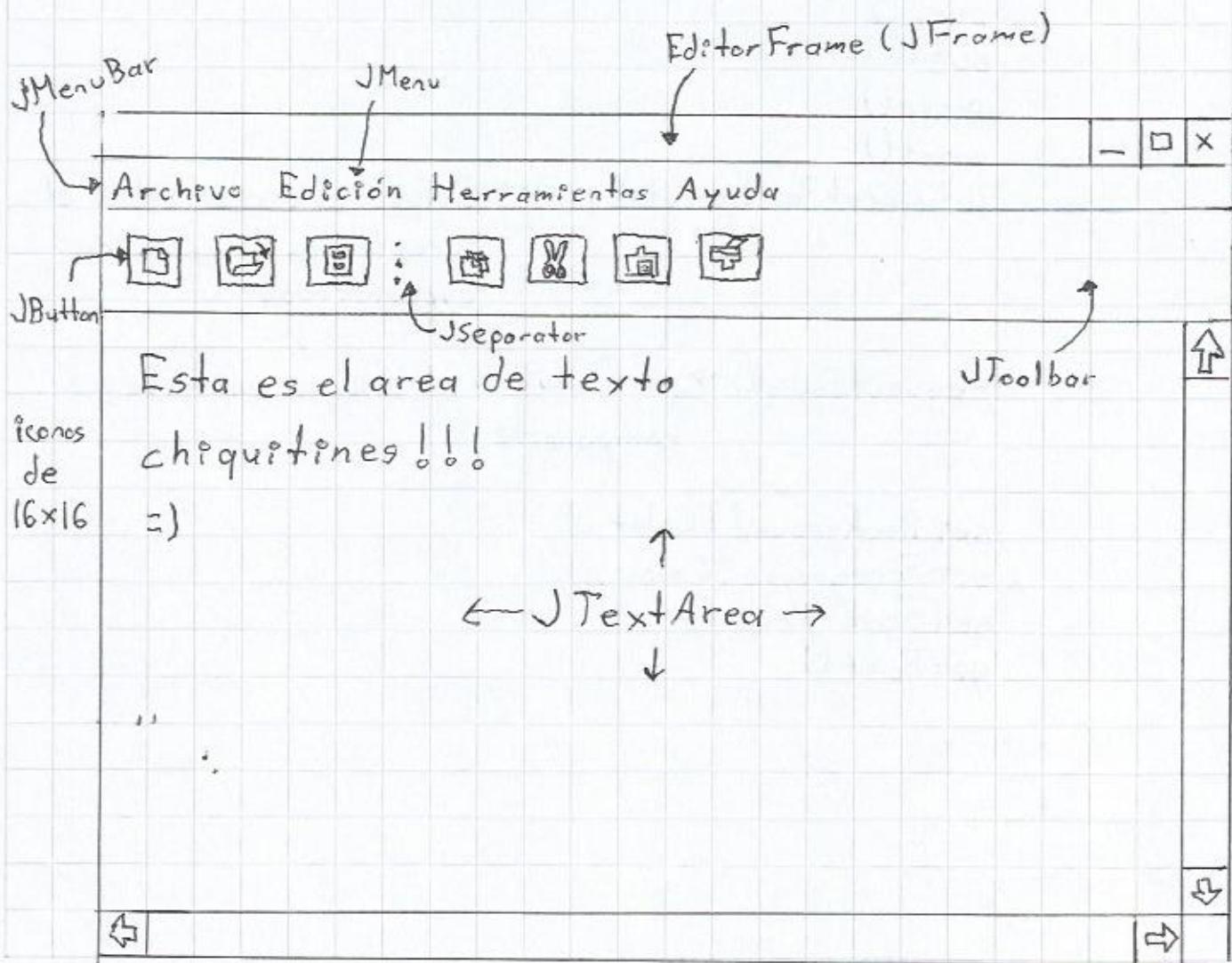
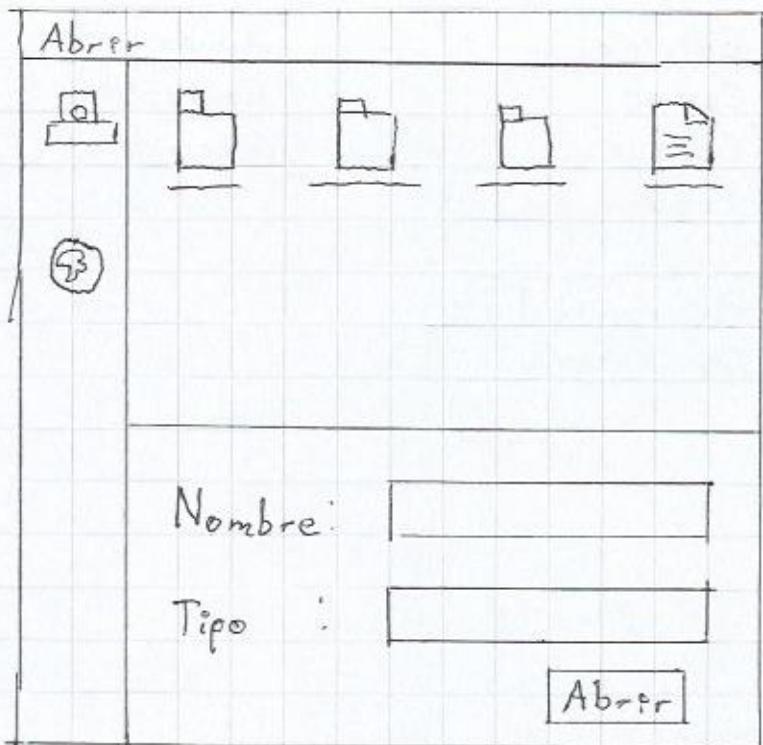
Basaremos el editor en un JTextArea que implementa funciones básicas de edición de texto plano, tales como: imprimir, copiar, pegar, cortar, manejar fuentes.

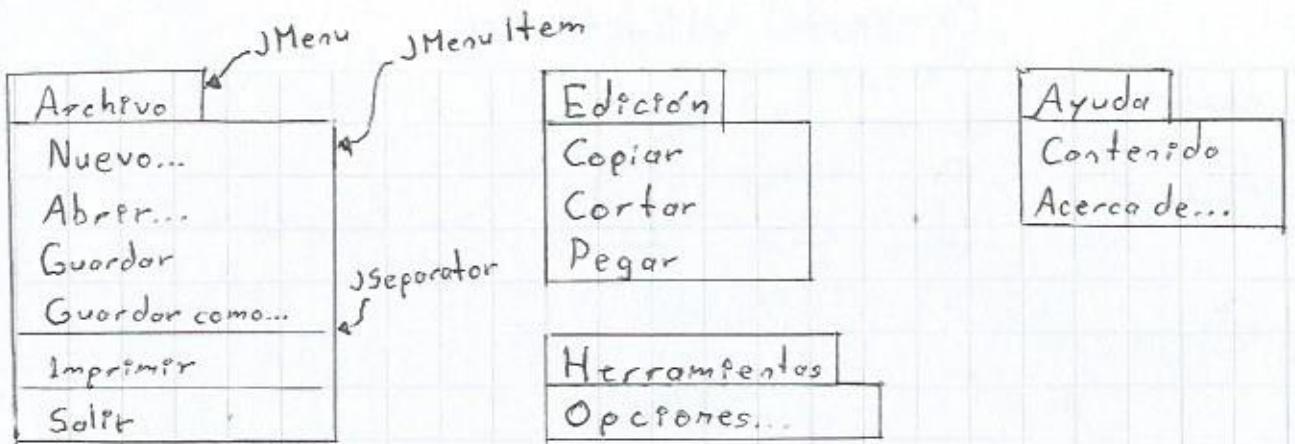
También se basará en manejo de archivos de texto (character streams) mediante clases tales como:

- File, FileReader, FileWriter, BufferedReader

Aprovecharemos la clase JFileChooser para mostrar diálogos standar "Abrir" y "Guardar como"

# Proyecto: UI Editor App





26- Febrero-2020

Clase: `JTextArea`

Métodos:

`getText()`  
`setText(String s)`  
`copy()`  
`cut()`  
`paste()`  
`print()`

`setCaretPosition (int pos)` → Coloca el cursor sobre el carácter en la posición especificada

`requestFocus()` → para ubicar el enfoque en este componente

`setBackground (Color c)`  
`setForeground (Color c)`  
`setFont (Font f)`  
`getFont()`

Clase: JFileChooser

Métodos: JFileChooser()  
showOpenDialog()  
showSaveDialog()

JFrame

setExtendedState(int estado)

NORMAL  
ICONIFIED  
MAXIMIZED\_BOTH

JDialog

Opciones

Apariencia Fuente Corrector

JPanel

Color

JComboBox

JLabel

Fondo: Blanco



Texto:

Negro

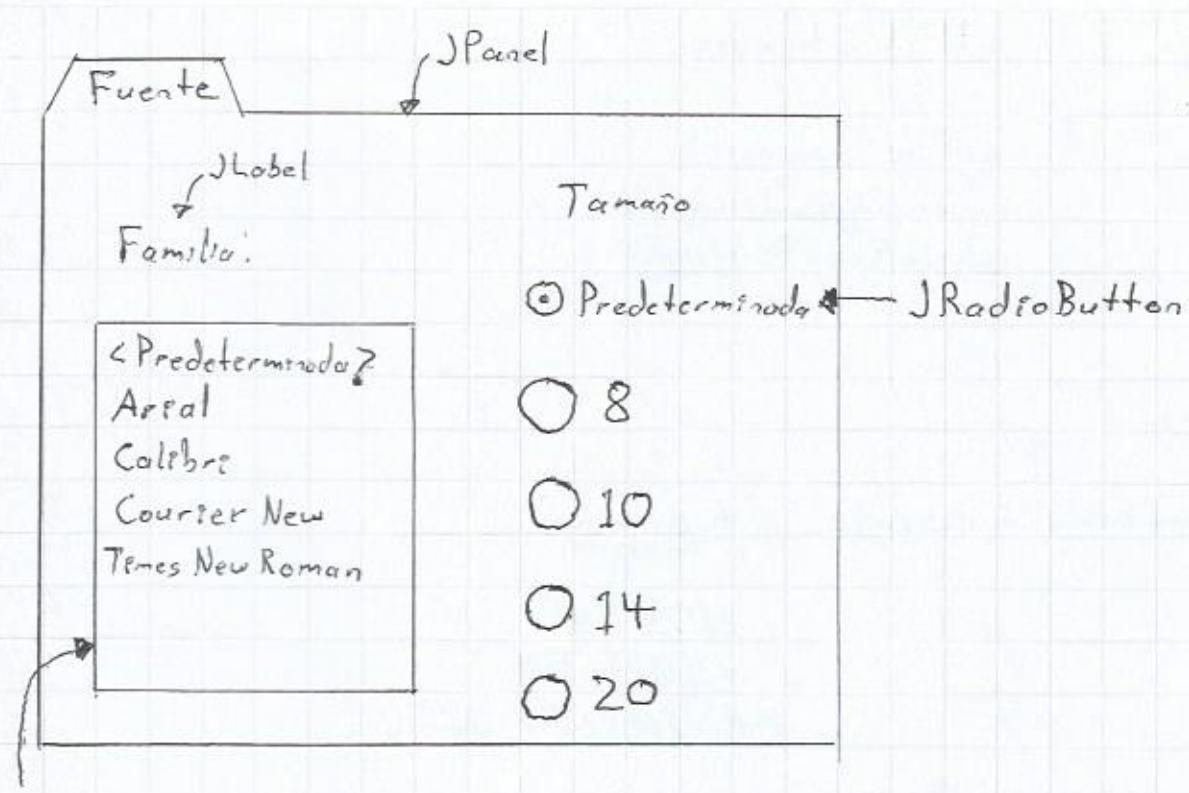


JTabbedPane

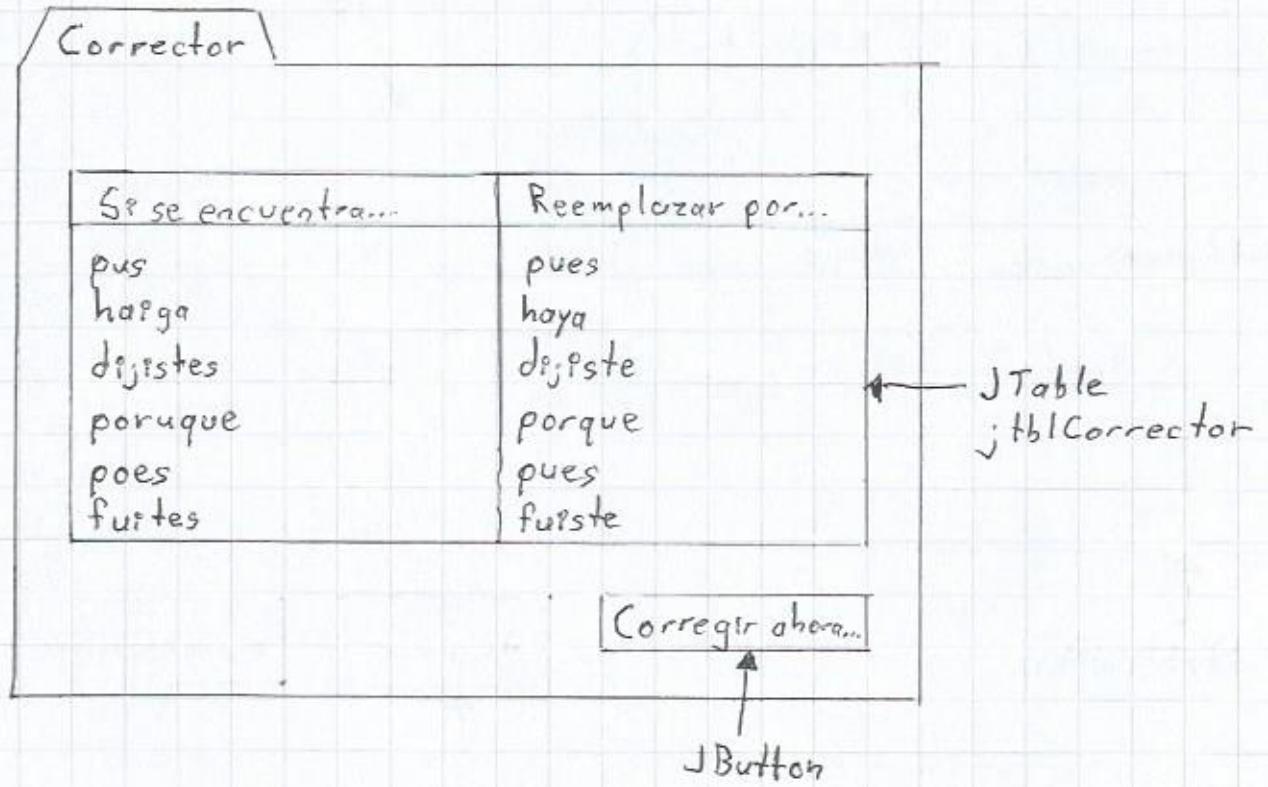
Aceptar

Cancelar

JButton



JList



## JTable

getRowCount()  
getValueAt(int row, int col)

## String

substring()  
indexOf()

03 - Marzo - 2020

Abrir un archivo pdf desde una aplicación Java

⇒ Clase: Desktop

Abrir el navegador con una URL dada desde una aplicación Java

\* Clase: Desktop

Método: browser(URI url)  
open(?)

Abrir otra aplicación o programa desde una aplicación Java

⇒ Clase: Process

```
java.awt.Desktop desktop = java.awt.Desktop.getDesktop();
if (desktop.isSupported (java.awt.Desktop.Action.BROWSE))
{
    try {
        String uriPath = "https://google.com";
        java.net.URI uri = new java.net.URI (uriPath);
        desktop.browse (uri);
    } catch (IOException e) {
        e.printStackTrace ();
    } catch (URLSyntaxException e) {
        e.printStackTrace ();
    }
} else {
    // Advertir al usuario que no soporta este accion
}
```

---

```
Process proceso = Runtime.getRuntime ()
    .exec ("C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe W:\\Archivo.pdf");
```

VI Miércoles 11 Marzo

- Traer su USB de la materia con sus programas

04-Marzo-2020

## Unidad II

### "Creación de librerías y componentes"

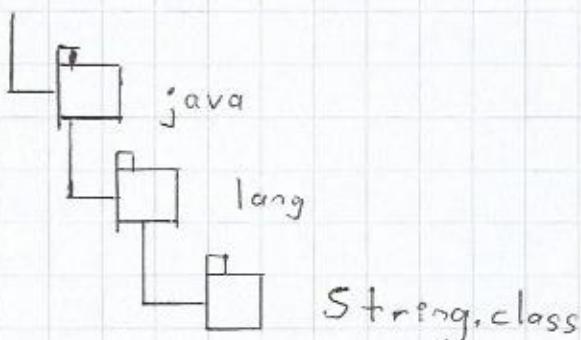
→ Librería: Colección de clases, interfaces y recursos a los que se hace referencia en proyectos nuevos.  
En Java las librerías se empacan y distribuyen en archivos .jar = Java Archive

jar {  
    = Librería  
    = Aplicación empaquetada

rt.jar = runtime

java.lang = package de default  
se carga automáticamente.

rt.jar



Proyecto

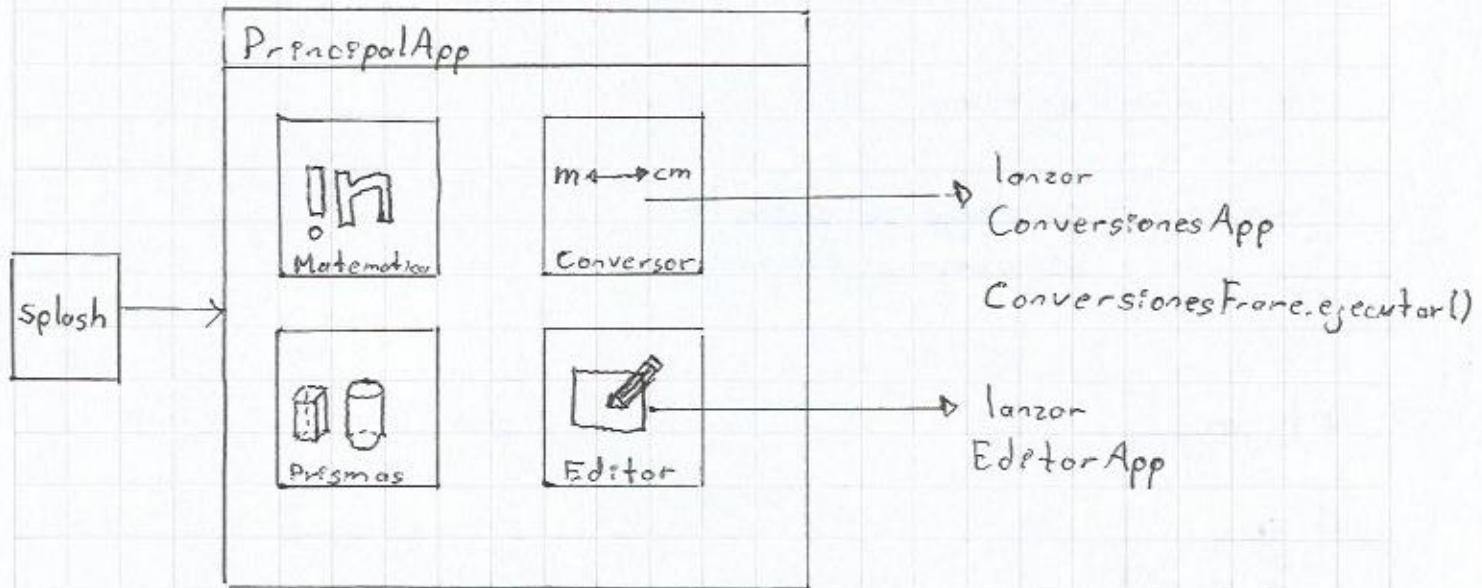
U2 TAPLib

JDK = Kit de Desarrollo

Software que debe tener el usuario final

JRE = Java Runtime Environment

[www.java.com](http://www.java.com)



10 - Marzo - 2020

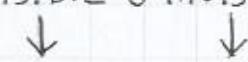
## Componentes

Es una pieza de software reutilizable, que tiene una funcionalidad encapsulada.

Consta de:

- Propiedades (personalizar en Tiempo de Diseño)
- Métodos (para modificar propiamente en tiempo de ejecución o invocar a su funcionalidad)
- Eventos (para notificar a otros componentes un cambio de estado)

Un componente puede ser visible o invisible



con IU    no tiene IU

Visible: Botón, Label, Panel de fichas, etc.

Invisible: Timer, ButtonGroup

En Java se les conoce como: JavaBeans o Beans

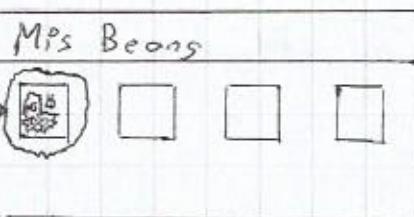
Mi componente



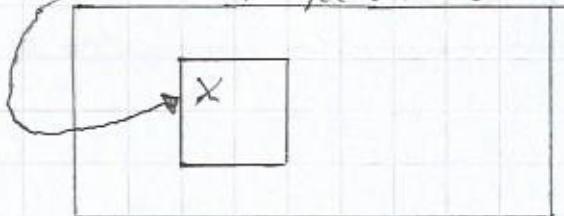
Logo del TEC

(TAPLib.jar)

Paleta de componentes



Proyecto Nuevo



Propiedades	
Prop 1	---
Prop 2	---
Prop 3	---

Un componente puede ser tan simple como una clase.java

Debe tener constructor de default

Ofrecer métodos setter/getter para las propiedades

Ej. Círculo, Cilindro, etc.

13 - Marzo - 2020

### BeanInfo

Es una clase java que se asocia a un bean y que personalizar información acerca del bean, tal como su ícono, propiedades preferidas, leyendas descriptivas, etc.



Bean.java



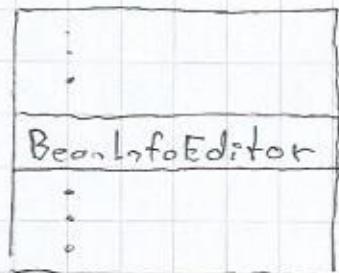
BeanBeanInfo.java

Circulo.java

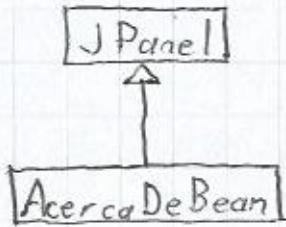
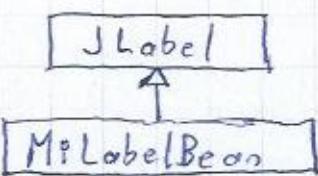
CirculoBeanInfo.java

Circulo.java

...  
clic derecho  
...



18 - Marzo - 2020



-- AcercaDeBean --

1 Tec Nac de Mex 13  
1 ITL 4  
1 J. S. C. s  
1 Topicos Av 6  
1 Autores:  
1 Futura? x<sup>2</sup>xxxx  
1 Editor? vLO<sup>10</sup>  
1 (c) Derechos Reservados, U\_

AcercaDeBean

JPanel

19-Marzo-2020

Logo  
TecNM

jlblInstitucion

→ TECNOLOGICO NACIONAL DE MEXICO

jlblLegal

→ Instituto Tecnológico de La Laguna

jlblCampus

jlblCarrera

Logo  
Campus

Ingenieria en Sistemas Computacionales

jlblMateria

TOPICOS AVANZADOS DE PROGRAMACION

jlblLogo2

→ [Nombre de la aplicación]

jlblNombreApp

V1.0 ← jlblVersionApp

→ Desarrollado por:

jlblDesarrollador

jlblAutor1 → José Misael Adame Sandoval

jlblControl1 → 18131209

jlblAutor2 →

jlblControl2 →

jlblAutor3 →

jlblControl3 →

c) Derechos Reservados 2020

jlblDerechosReservados

23-Marzo-2020

JPop Up Menu

Al clickear "Acerca de..." aparece el diálogo

→ Acerca de...

mn: AcercaDe

Acerca de...

X



AcercaDeBean v.1.0

Por

José Misael Adame Sandoval (18131209)

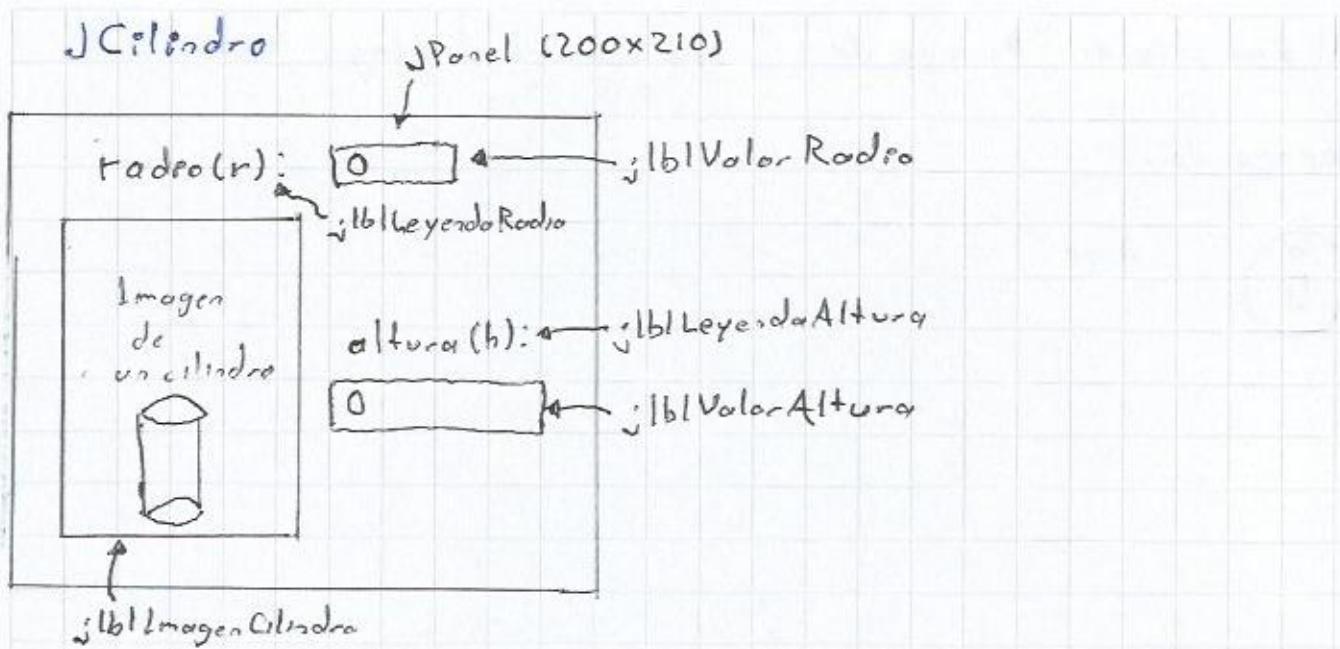
Semestre: Ene-Jun/2020

[Aceptar]

Todos los componentes del Bean pueden ser modificados (labels) en tiempos de diseño.

En el jlblLogo1 y jlblLogo2 se añade que se pueda ajustar la imagen con el método escalarImagen() de la Clase Imágenes mediante el uso de un jCheckBox.

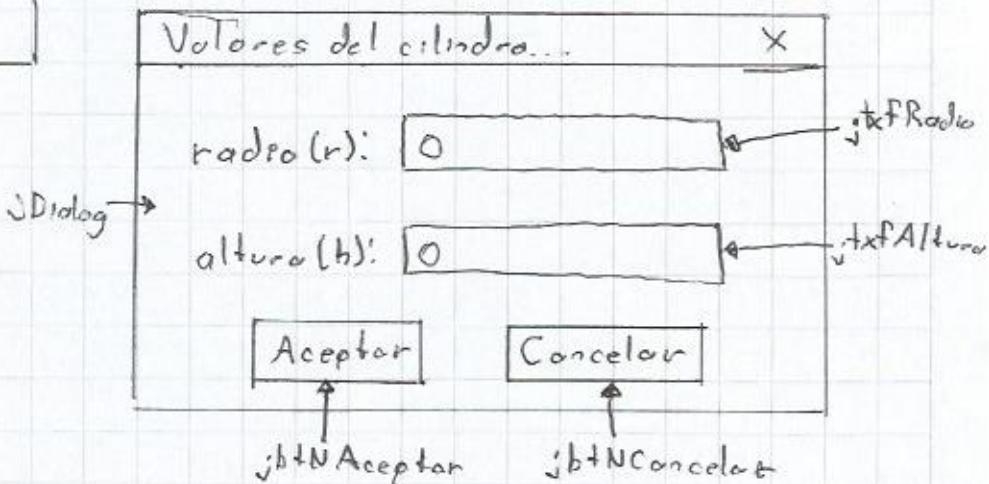
25 - Marzo - 2020



JPopupMenu

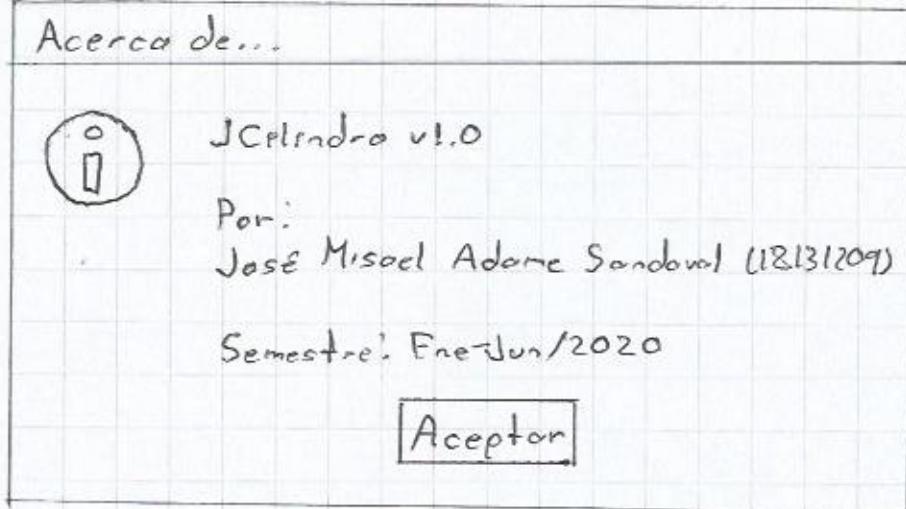
Valores del prisma...  
Acerca de...

Al dar clic en "Valores del Prisma..." aparece el siguiente JDialog Cilindro



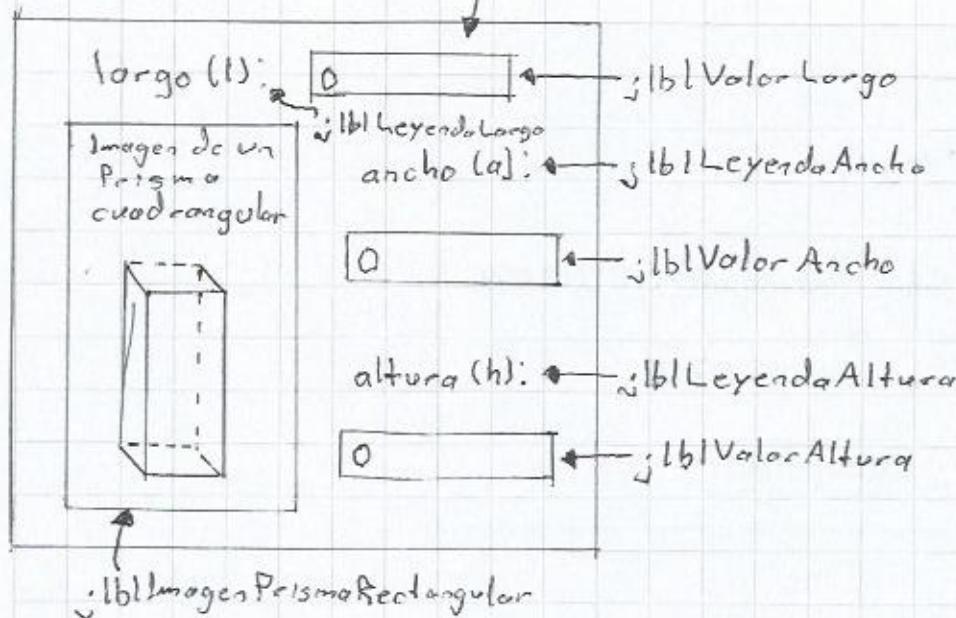
Todos los componentes del JCilindro pueden ser modificados (labels) en tiempo de diseño

Al dar clic en "Acerca de..." aparece el diálogo



En el `JlblImagenCilindro` se añade una propiedad para que se pueda ajustar la imagen con el método `escalarImagen()` de la Clase `Imagenes` mediante el uso de un `JCheckBox`.

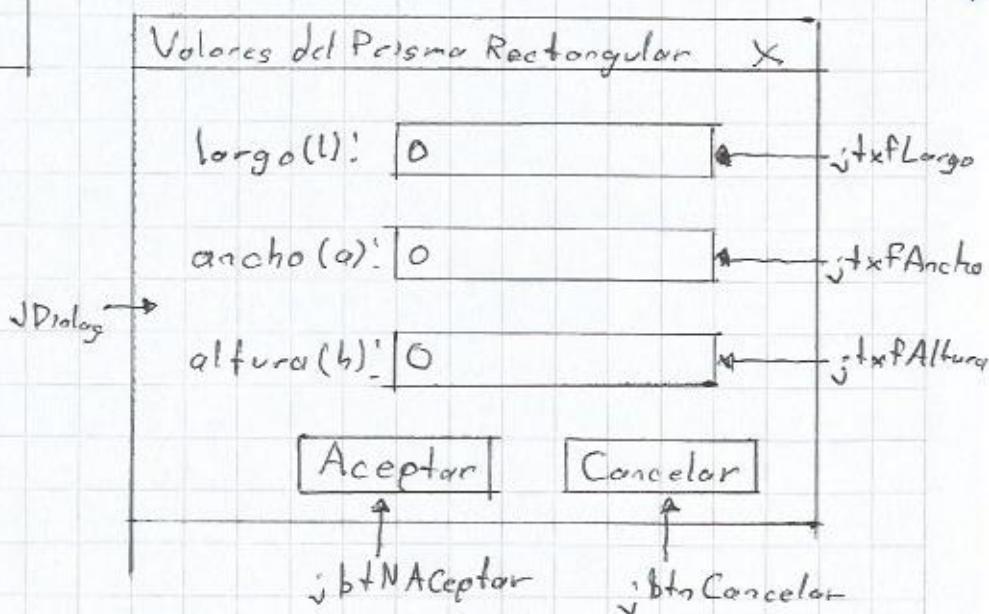
## JPrismaRectangular JPanel (200x210)



## JPopUpMenu

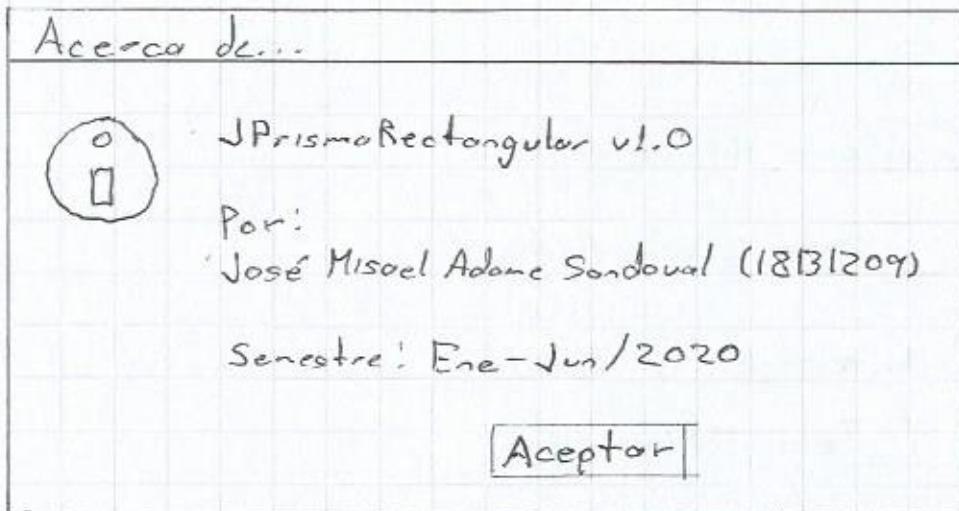
Valores del prisma...  
Acerca de...

Al dar click en "Valores del Prisma..." aparece el siguiente JDialogPrismaRectangular



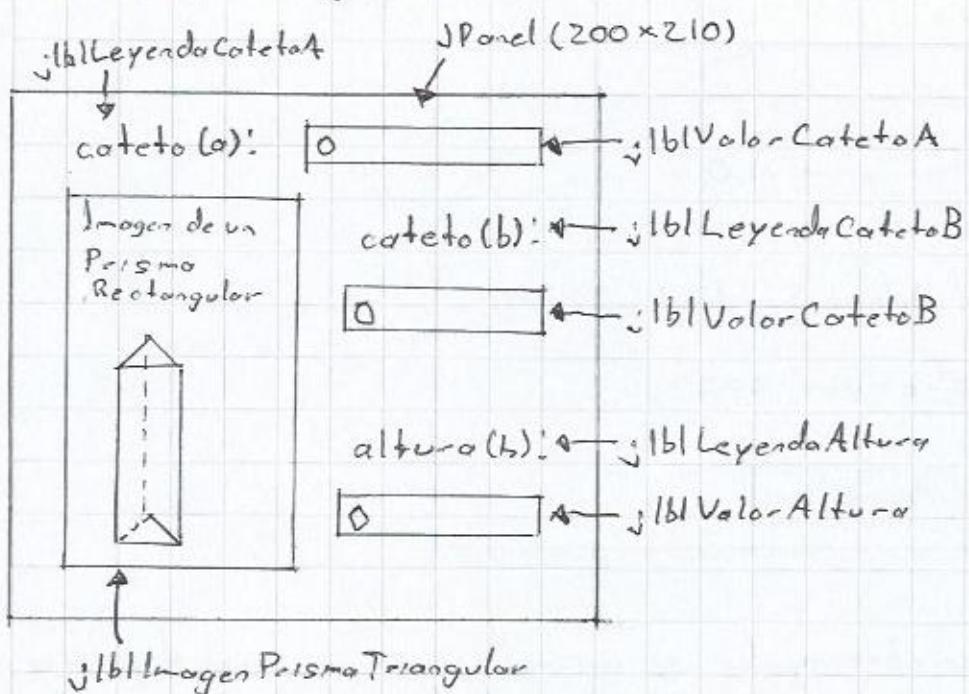
Todos los componentes del JPrismaRectangular pueden ser modificados (labels) en tiempo de diseño

Al dar clic en "Acerca de..." aparece el diálogo



En el `JlblImagenPrismaRectangular` se añade una propiedad para que se pueda ajustar la imagen con el método `escalarImagen()` de la clase `Imagenes` mediante el uso de un `checkbox`.

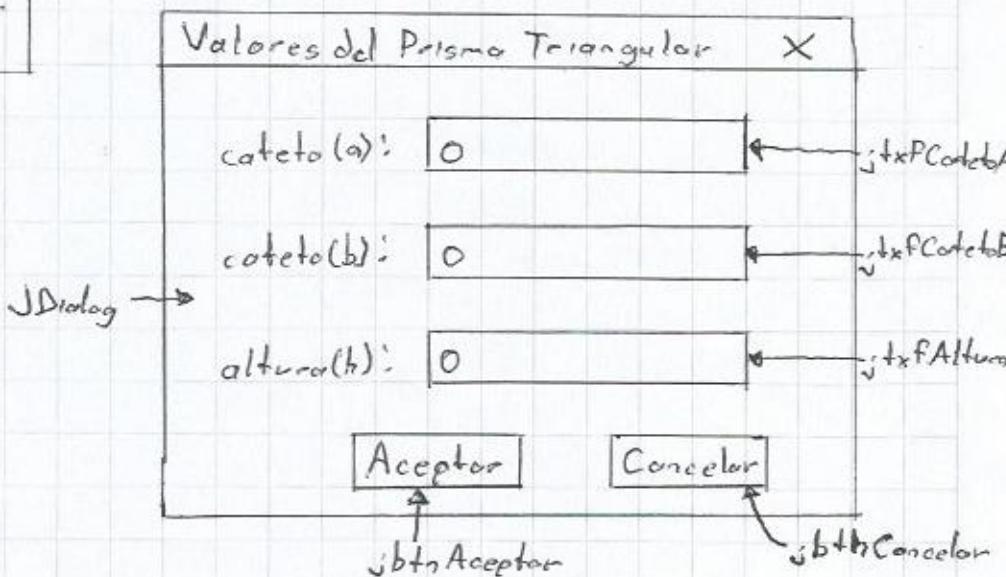
## JPrismaTriangular



## JPopupMenu

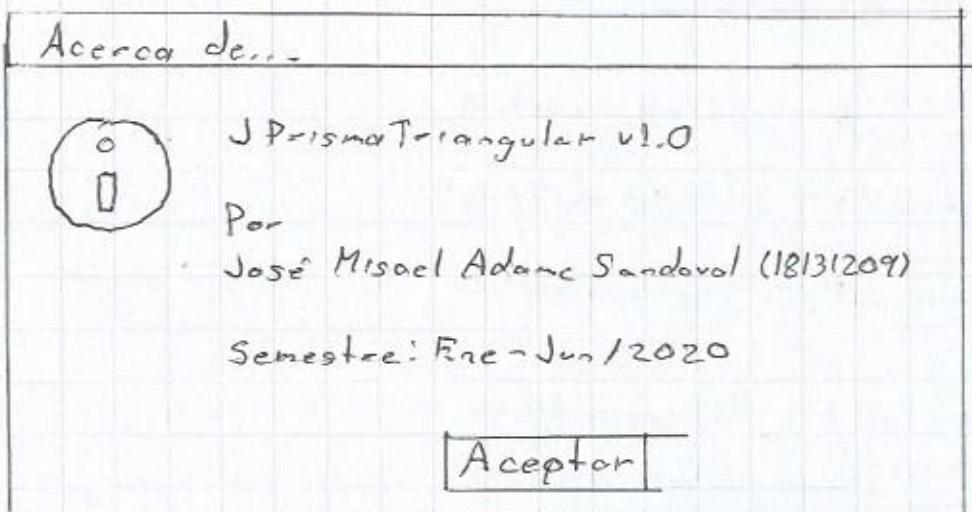
Valores del prisma...  
Acerca de...

Al dar clic en "Valores del Prisma..." aparece el siguiente JDialogPrismaTriangular



Todos los componentes del JPrismaTriangular pueden ser modificados (labels) en tiempo de diseño

Al dar clic en "Acerca de..." aparece el diálogo



En el JPanelImagenPrismaTriangular se añade una propiedad para que se pueda ajustar la imagen con el método escalarImagen() de la clase Imagenes mediante el uso de un CheckBox.

## Introducción

Los eventos son notificaciones que el Bean es capaz de hacer a otros Beans sobre algo sucedido con él, por ejemplo, un cambio de estado. Un cambio de estado puede ser simplemente el cambio de valor en una de las variables del Bean. No solo en cambios de estado se disparan eventos, un Bean puede disparar un evento al iniciar una tarea y otro al terminar dicha tarea; de manera que un Bean puede tener toda una serie de eventos que sea capaz de disparar en diferentes momentos o condiciones. Los eventos que un Bean genera son recibidos por otros Beans, llamados escuchantes o más comúnmente "listeners", los listeners al recibir la notificación de un evento desde otro Bean podrán procesar dicho evento de acuerdo a lo que su funcionalidad les dicte.

La programación visual (GUI) que también es conocida como programación conducida por eventos (event driven programming) se basa en los conceptos citados anteriormente. Un JButton, un JList, un JComboBox son componentes visuales que pueden disparar diferentes eventos. Pero no solo los Beans visibles disparan eventos también puede ser un bean invisible, por ejemplo, un Timer que cada vez que se cumple su cuenta de tiempo es capaz de disparar un evento.

El entendimiento del mecanismo de generación de eventos en Java es muy similar al Patrón de Diseño OBSERVER. En este patrón de diseño hay una clase llamada Observable y varias clases llamadas Observer. Los Observer deben suscribirse con el Observable para poder recibir la notificación de un evento de interés que el Observable genere. Si un Observer no se suscribe con el Observable nunca recibirá el evento de interés.

## Modelo de Eventos de Java

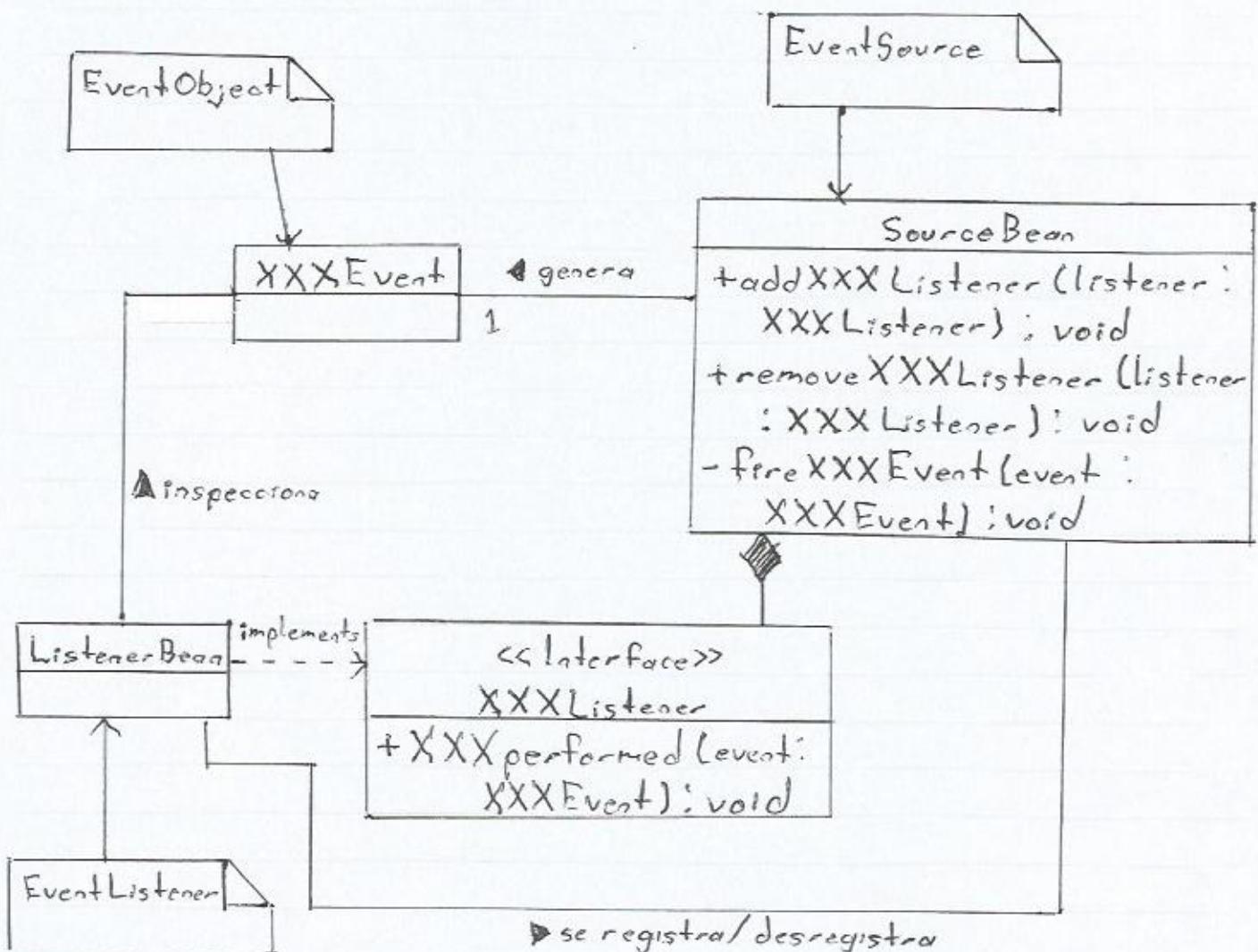
También conocido como Modelo de Delegación de Eventos detalla el mecanismo de funcionamiento para que un Bean genere eventos y los Listeneres sean capaz de recibirlas. En este modelo participan 3 elementos:

1. EventSource o Source. - Es el Bean que genera el evento.
2. EventListener o Listener. - Es un bean que previamente se ha registrado para poder escuchar el evento desde el EventSource.
3. EventObject. - Un objeto que encapsula información relativa al evento.

## Modelo de eventos en la implementación

La siguiente figura muestra de forma general las clases y métodos que deben definirse para implementar el modelo de eventos de Java.

Los símbolos de hoja blanca señalan a las clases que representan cada una de los participantes en el modelo: EventSource, EventObject y EventListener.



- La clase **SourceBean** representa a cualquier clase que sea un JavaBean.
- La clase **Listener Bean**, esta clase representa a cualquier clase Java.
- En el diagrama se debe sustituir la **XXX** generalmente por el nombre del evento.

21-Abril-2020

Unidad III

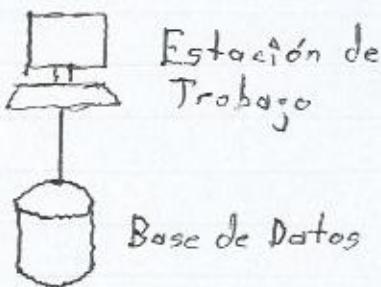
Conectividad con Bases de Datos

JDBC

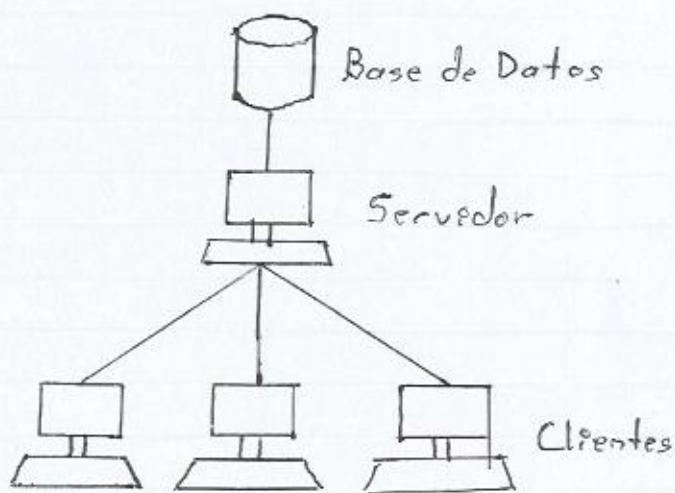
Java DataBase Connectivity

Parte de la API JavaSE que proporciona clases e interfaces para conectar a diferentes bases de datos, consultarlos y actualizarlos.

BD local



BD remota



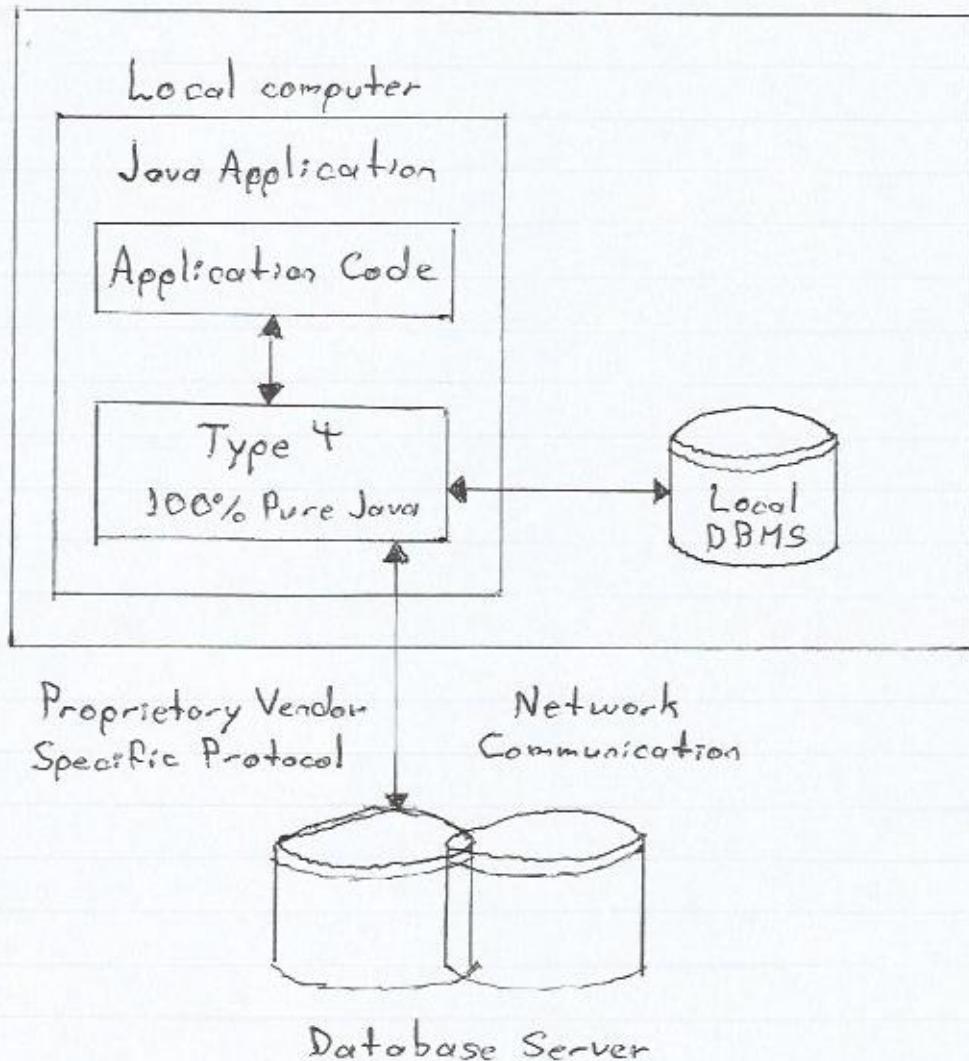
- Bases de datos tipo SQL, ej. SQL Server, Oracle, MySQL, Access, etc.
- JDBC se encuentra en el paquete `java.sql`
- Requisito: Controlador JDBC compatible (driver)

## Controladores JDBC

- Tipo I, II, III, IV (tipo IV es el mejor)
- Driver Tipo IV: 100% Pure Java Driver
- Cada fabricante de BD ofrece su controlador JDBC tipo IV.  
Ej. Microsoft ofrece su driver JDBC para SQL Server
- El controlador se distribuye en archivos .JAR

## Controlador JDBC Tipo IV

Se necesita estrictamente el controlador tipo IV para que esta pueda comunicarse entre el código de la aplicación y la base de datos, ya sea local o remota.



23 - Abril - 2020

## Conexión a Java DB por código

Información necesaria:

1. Librerías JAR del controlador JDBC
2. Nombre de la clase del Driver JDBC
3. URL de conexión
4. Nombre de la base de datos
5. Usuario
6. Contraseña

### Requerimientos

Librerías JAR del controlador JDBC =

/Library/Java/JavaVirtualMachines/jdk1.8.0\_111.jdk/Contents/Home/db/lib/derbyclient.jar

/Library/Java/JavaVirtualMachines/jdk1.8.0\_111.jdk/Contents/Home/db/lib/derby.jar

driver JDBC = org.apache.derby.jdbc.ClientDriver

URLConexión = jdbc:derby://localhost:1527/TAPdb

Usuario = topicos

Contraseña = 2020

## Código

Cargar el driver:

```
Class.forName(driverJDBC);
```

Crear el objeto Connection:

```
Connection con = DriverManager.getConnection(  
    URLConexion + nombreBD, usuario, contraseña);
```

```
if (con != null) {  
    // Conexión exitosa  
}
```

// ResultSet = Conjunto de registros que devuelve la consulta

Al terminar de usar la conexión:

```
con.close();
```

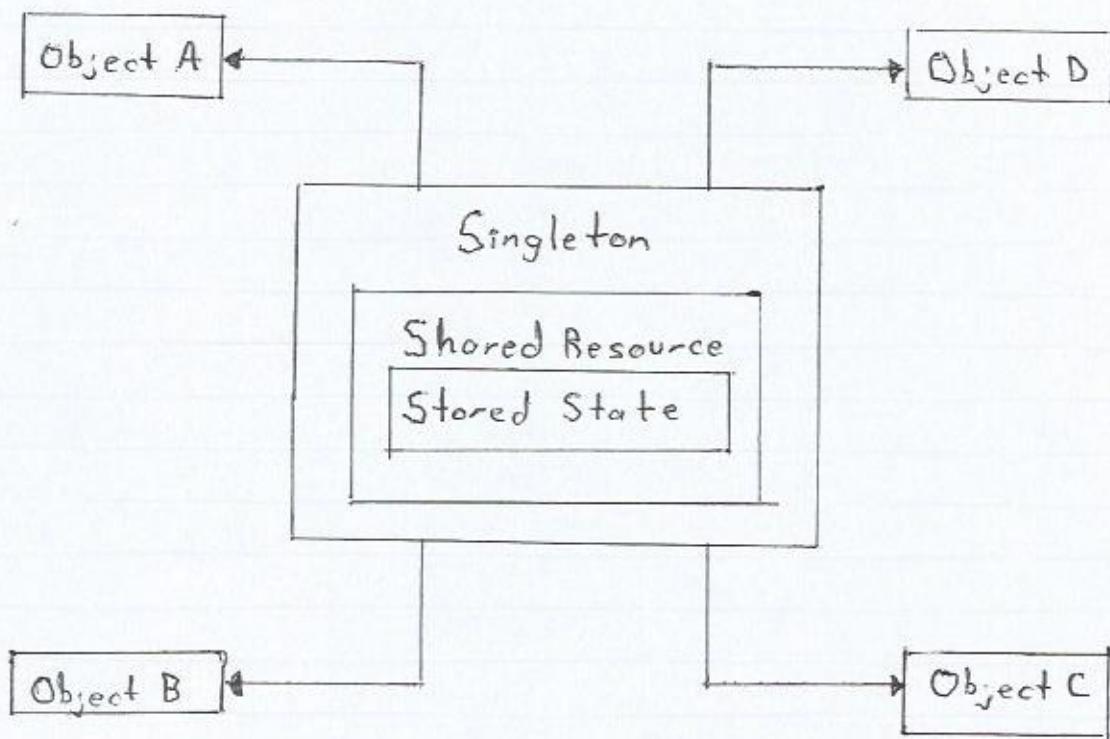
Atrapar las excepciones

```
ClassNotFoundException
```

```
SQLException
```

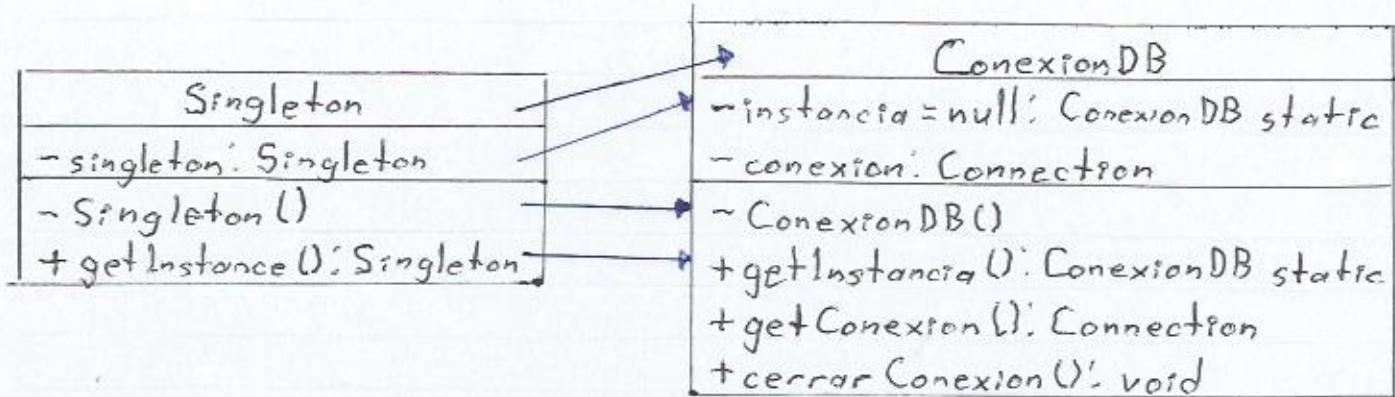
27-Abril-2020

Objeto de Conexión basado en patrón Singleton



Clase Singleton:

- Declara una variable privada de la propia clase Singleton
- Constructor de default se declara privado
- La única forma de obtener la instancia valida del Singleton es mediante el método getInstance().



`private ConexionDB() {`

`// Crear una conexión JDBC y guardarla en la variable conexión  
}`

`public static ConexionDB getInstance() {`

`if (instancia == null)  
 instancia = new ConexionDB();  
return instancia;`

`}`

Obtener el objeto de conexión:

`Connection con = ConexionDB.getInstance().getConexion();`

Cerrar la conexión:

`ConexionDB.getInstance().cerrarConexion();`

22-Mayo-2020

## Unidad IV Programación Concurrente Multihilo

### Programación concurrente

Concurrencia se da cuando dos o más tareas se desarrollan en el mismo intervalo de tiempo.

Unidades básicas de ejecución en programación concurrente:

Procesos es un programa en ejecución. Tiene su propio espacio de memoria, enlaces a recursos, etc.

Threads es un camino de ejecución dentro de un proceso.  
También se les llama subprocesos o subtareas.  
Cada proceso tiene al menos un hilo, llamado hilo principal.

### División de Tiempo

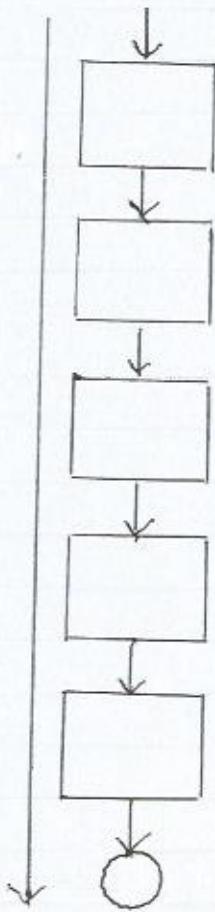
Una computadora tiene normalmente muchos procesos e hilos activos. Esto es cierto aun en sistemas que tienen un solo núcleo de procesador en el que solo un hilo se podría ejecutar en un momento dado.

El tiempo de procesamiento para un único núcleo de procesador es compartido entre procesos e hilos a través de una característica del SO llamada división de tiempo (time slicing).

Hoy día los procesadores con múltiples núcleos mejoran enormemente la capacidad de un sistema para la ejecución concurrente de procesos e hilos.

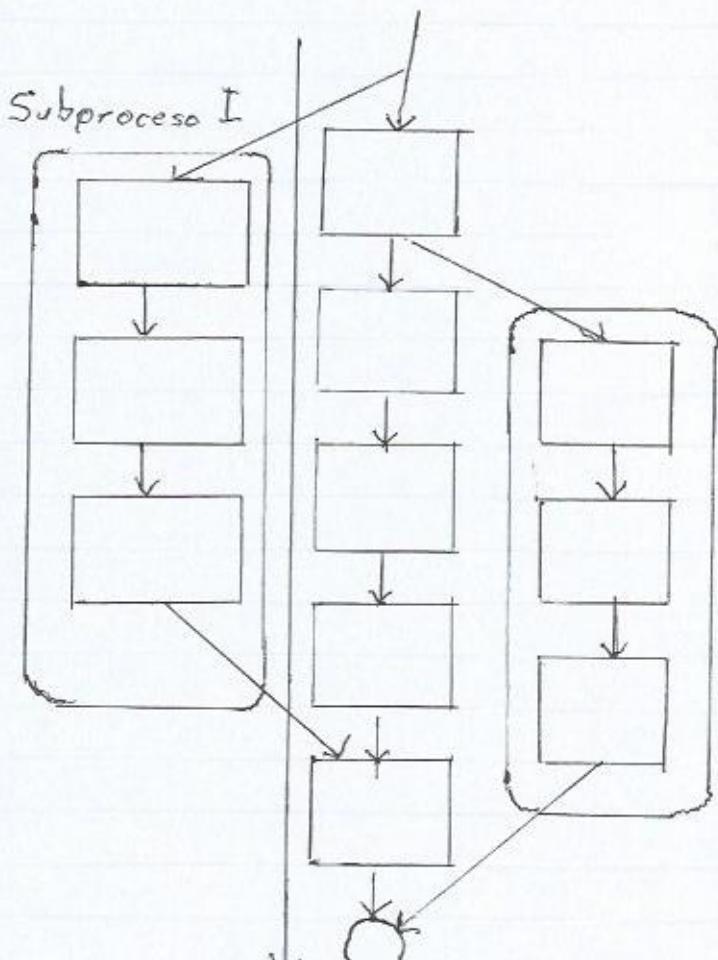
# Monotarea vs Multitarea

Proceso monotarea



Camino de ejecución principal

Proceso multitarea



Camino de ejecución principal

## Creación de hilos con Thread

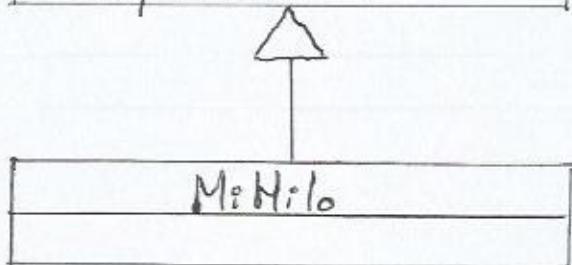
Crear una subclase de:

Thread

Sobreescibir el método:

run()

Thread
+Thread()
+Thread(nombre; String)
+Thread(runnable; Runnable)
+run()
+start()
+stop()
+pause()
+resume()
+sleep(milliseconds)



En el método run() se programa la tarea que va a realizar el hilo

public class MiHilo extends Thread

{@Override}

public void run () {

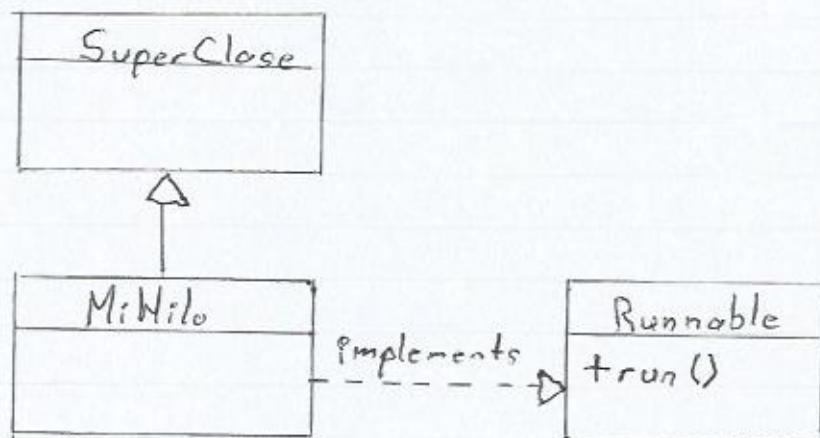
//implementar aqui la tarea

}

## Creación de hilos con Runnable

Implementando el interface: Runnable

Implementar el método: run()



```
public class MiHilo extends SuperClose
{
    implements Runnable
    public void run () {
        // implementar aquí la tarea
    }
}
```

## Ejecución de hilos con Thread

Un Thread se ejecuta invocando al método start()

Nunca se debe invocar a run() directamente.

```
MiHilo hilo1 = new MiHilo();  
hilo1.start();
```

start() automáticamente invoca a run()

## Ejecución de hilos con Runnable

Un hilo basado en el interface Runnable se ejecuta indirectamente con un objeto Thread:

```
MiHilo hilo1 = new MiHilo();  
Thread thread = new Thread(hilo1);  
thread.start();
```

Al invocar start() del objeto thread invocará automáticamente al método run() del objeto hilo1.

## Terminación de un hilo

```
public void run()
```

...

```
}
```



Un hilo termina (muere) al alcanzar la llave de cierre del método run().

No se puede invocar start() de un hilo que ya ha terminado, se debe crear una instancia nueva.

## Dormir un hilo

En un hilo se programan normalmente tareas que tardan cierto tiempo o carga de trabajo ejecutarlas.

Para que un hilo no ocupe el procesador y permita a otros hilos ejecutarse es necesario "dormir" el hilo por un tiempo determinado mediante el método estático sleep():

```
public void run()
```

```
}
```

Thread.sleep(500); // 0.5 seg