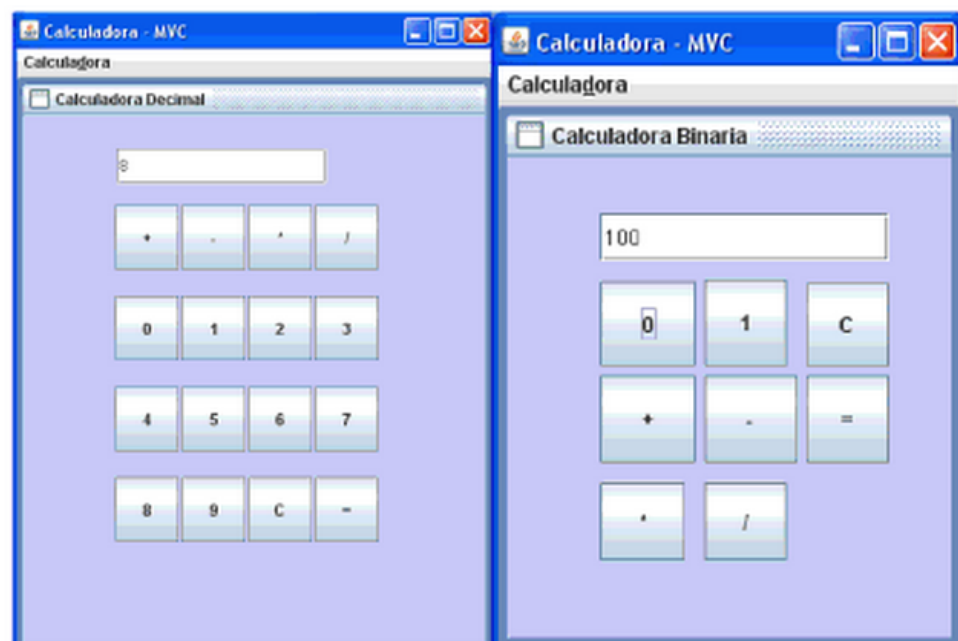


### 3. ARQUITECTURA MVC (MODEL-VIEW-CONTROLLER), ESTRUCTURAS DE ACCESO DIRECTO POR LLAVE Y XML

#### 3.1. Caso de estudio 1 Calculadora

Se desea crear una aplicación donde se muestre el patrón MVC. Se manejarán 2 tipos de vistas-controlador las cuales serán observadores de una calculadora. Esto quiere decir, que cuando la información de la calculadora se modifique entonces las vistas-controlador deberán efectuar la modificación pertinente.

Los tipos de vistas-controlador que debe manejar la aplicación son: (1) Vista-controlador Decimal, la cual muestra el número ingresado o producto de un cálculo en decimal. Este controlador debe permitir cambiar los valores de los números que serán operados. Es de anotar que la calculadora manejará solo las operaciones de suma, resta, multiplicación y división. (2) Vista-controlador Binario, que permite mostrar los números ingresados o producto de un cálculo en formato binario. El controlador debe permitir cambiar el número en formato binario.



#### 3.1.1 Comprensión del problema

##### a) Requerimientos funcionales

NOMBRE	R1 – Crear vista controlador decimal
RESUMEN	Se crea una vista-controlador que muestre al número ingresado o producto de un cálculo en formato decimal.
ENTRADAS	

<b>RESULTADOS</b>
Se creó una vista-controlador que observa la información ingresada o producto de un cálculo de la calculadora y lo muestra en decimal.

<b>NOMBRE</b>	R2 – Crear vista controlador binario
<b>RESUMEN</b>	Se crea una vista-controlador que muestre al número ingresado o producto de un cálculo en formato binario.
<b>ENTRADAS</b>	
<b>RESULTADOS</b>	
	Se creó una vista-controlador que observa la información ingresada o producto de un cálculo de la calculadora y lo muestra en binario.

<b>NOMBRE</b>	R3 – Fijar un nuevo operando para la operación que se va a realizar en decimal.
<b>RESUMEN</b>	Se ingresa un nuevo operando en formato decimal.
<b>ENTRADAS</b>	
	Nuevo número en decimal
<b>RESULTADOS</b>	
	Se modificó el valor del nuevo operando.

<b>NOMBRE</b>	R4 – Fijar un nuevo operando para la operación que se va a realizar en binario.
<b>RESUMEN</b>	Se ingresa un nuevo operando en formato binario.
<b>ENTRADAS</b>	
	Nuevo número en binario
<b>RESULTADOS</b>	
	Se modificó el valor del nuevo operando.

<b>NOMBRE</b>	R5– Realizar la operación especificada.
<b>RESUMEN</b>	Se realiza la operación elegida entre los dos últimos operandos ingresados
<b>ENTRADAS</b>	
<b>RESULTADOS</b>	
	El resultado de la operación

### 3.1.2 Arquitectura MVC (Modelo Vista Controlador)

Fue creada en 1979 por Trygve Reenskaug. Es un patrón que permite separar la GUI, de los datos y de la lógica apoyándose en tres componentes. A saber:

- **Modelo:** Esta es la representación de los datos y reglas de negocio (mundo del problema). Es el encargado de manejar un registro de las vistas y de los controladores que existen en el sistema.
- **Vista:** Permite mostrar la información del modelo en un formato adecuado que permita que se de la interacción. Además de poseer un registro acerca del controlador asociado y brinda el servicio de update que puede ser usado tanto por el controlador como por el modelo.
- **Controlador:** Responde a los eventos provocados por el usuario (se da un clic, se digita un texto, etc ) que implican cambios en el modelo y la vista, dando una correcta gestión a las entradas del usuario.

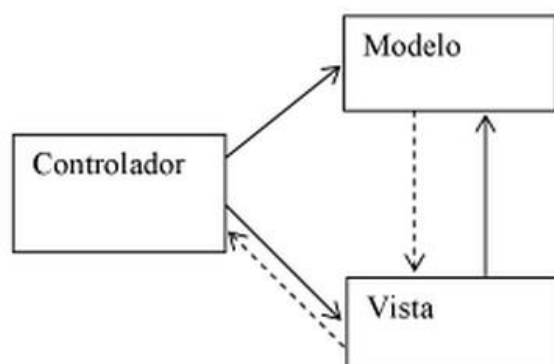


Figura 3.1. Relación entre el controlador, vista y modelo, la línea punteada indica la implementación del patrón Observer.

Es bien sabido que en aplicaciones a gran escala lo que cambia con mayor frecuencia es la lógica, por ello un diseño incorrecto en el cual se fusionen la interfaz de usuario con el mundo conllevaría a que fuera necesario modificar varios elementos del negocio en el caso en que se presentase una necesidad de transformación, conllevando a mayores gastos y riesgos.

Por ello se requiere que se desacople la vista, logrando con ello mayor reusabilidad.

Java brinda soporte a esta arquitectura a través de Observer y Observable. La primera clase permite que el objeto sea informado cuando cambie el estado de otro objeto. Mientras que el segundo, es el objeto fuente de interés.

La Vista es un subtipo de Observer, mientras que el modelo es un subtipo de Observable.

```

public class Calculadora extends Observable
{
    .
    .
    .
}
  
```

```

public class PanelBinario extends javax.swing.JPanel implements Observer,
ActionListener
{
    .
    .
    .
}
  
```

Para el control de Observer y Observable pueden utilizarse funciones. En el caso de Observer se puede usar el método update, el cual es invocado en el momento en el que se efectúa un cambio en el estado del objeto que se observa.

```
public void update( Observable obs, Object obj )
```

A su vez Observable ofrece las siguientes operaciones:

- `public void addObserver( Observer obs )` - Agrega un observador.
- `public void deleteObserver( Observer obs )` - Elimina un observador del listado de observadores.
- `public void deleteObservers()` - Remueve todos los observadores.
- `public int countObserver()` - Indica la cantidad de observadores existentes en la lista interna.
- `protected boolean hasChanged()` - Retorna un true si el Observable ha cambiado de estado.
- `public void notifyObservers( Object obj )` - Comprueba el flag interno para ver si el Observable ha cambiado de estado y lo notifica a todos los observadores, en caso de que el flag interno de Observable haya cambiado de estado. De esta forma les envía el objeto argumento para que los observadores hagan uso de el a través de su método notify().

Jaramillo Valbuena, Sonia., Augusto Cardona, Sergio., Villa Zapata, Dumar Antonio. (2008).

*Programación avanzada en Java*. Armenia Quindío: Elizcom