Objetivos:

- Diseño OO, Patrones
- División en capas MVC, Swing
- Event-Handling

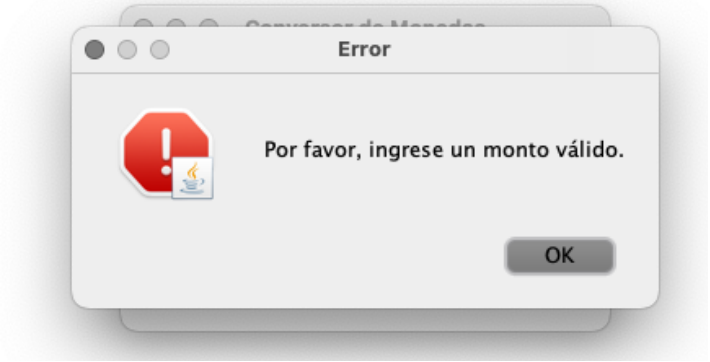
Ejercicio 1 – Conversor de Monedas

La agencia de cambio *Metrópolis* ha solicitado un sistema de conversión de monedas para la utilización de sus operaciones diarias. El sistema tendrá un *Look and Feel* similar a la figura 1 y permitirá la elección de las monedas involucradas en la transacción y la carga del monto a convertir.

Luego, de indicar estos datos se procederá a la obtención del monto que corresponde en la moneda seleccionada como destino. Es decir, la cotización de la moneda destino por el monto a cambiar.

Figura 1

En el caso de que el cálculo no se pueda realizar se deberá mostrar un diálogo de alerta similar a:



Ejercicio 2 – *Las Fases Lunares*

Se desea construir un programa (ver figura 2), llamado *Fases Lunares,* que permita ayudar a aquellas personas que necesitan conocer la fase de la luna en un momento dado. Es decir, nuestro programa permitirá cargar por pantalla una fecha y a partir de esta, calculará la fase en que se encontrará la luna.

Utilice MVC para separar el modelo (capa lógica o de negocios) , el cual es el encargado de realizar los cálculos y la IU (capa de presentación) que es la encargada de mostrar los mismos.

Los métodos utilizados para calcular la fase de la luna son los siguientes:

```
public class LunarPositionCalculator {

    // Constantes para cálculos lunares
    private static final double LUNAR_CYCLE_DAYS = 29.5305882;
    private static final double ADJUSTMENT_CONSTANT = 694039.09;

    // Enum para representar las fases lunares
    public enum Fase {
        LunaNueva,
        CrecienteIluminante,
        CuartoCreciente,
        GibosaIluminante,
        LunaLlena,
        GibosaMenguante,
        CuartoMenguante,
        CrecienteMenguante
    }

     //Determina la posición de una fecha en relación con las fases lunares
    private static int determinarPosicion(Date date) {

        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);

        // Extraer día, mes y año de la fecha proporcionada
        int day = calendar.get(Calendar.DATE);
        int month = calendar.get(Calendar.MONTH) + 1;

        // Se suma 1 porque en Calendar los meses van de 0 a 11
        int year = calendar.get(Calendar.YEAR);

        double c, e, jd = 0.0;
        int position = 0;

        // Ajustar la fecha si el día es menor a 3
        if (day < 3) {
            year--;
            month += 12;
        }

        // Calcular el valor jd relacionado con las fases lunares
        c = 365.25 * year;
```

```
        e = 30.6 * month;
        jd = c + e + day - ADJUSTMENT_CONSTANT;
        jd /= LUNAR_CYCLE_DAYS;

        // Redondear y ajustar el valor jd para obtener el resultado final
        position = (int) jd;
        jd -= position;
        position = (int) Math.round(jd * 8);

        // Si la posición es mayor o igual a 8, establecerla en 0
        if (position >= 8) {
            position = 0;
        }

        // Devolver la posición lunar calculada
        return position;
    }

    public static Fase calcularFase(Date date) {
        switch (determinarPosicion(date)) {
            case 0:
                return Fase.LunaNueva;
            case 1:
                return Fase.CrecienteIluminante;
            case 2:
                return Fase.CuartoCreciente;
            case 3:
                return Fase.GibosaIluminante;
            case 4:
                return Fase.LunaLlena;
            case 5:
                return Fase.GibosaMenguante;
            case 6:
                return Fase.CuartoMenguante;
            case 7:
                return Fase.CrecienteMenguante;
            default:
                throw new RuntimeException("Error al determinar la fase!");
        }
    }

public static void main(String[] args) {
        // Ejemplo de uso
        Date fechaEjemplo = new Date();
        // Puedes reemplazar esto con tu propia fecha

        Fase faseLunar = calcularFase(fechaEjemplo);
        System.out.println("La fase lunar para la fecha es: " + faseLunar);
    }

}
```

Nota: *En la **PlataformaEd** se adjuntan las imágenes que representan los distintos estadios de la luna.*

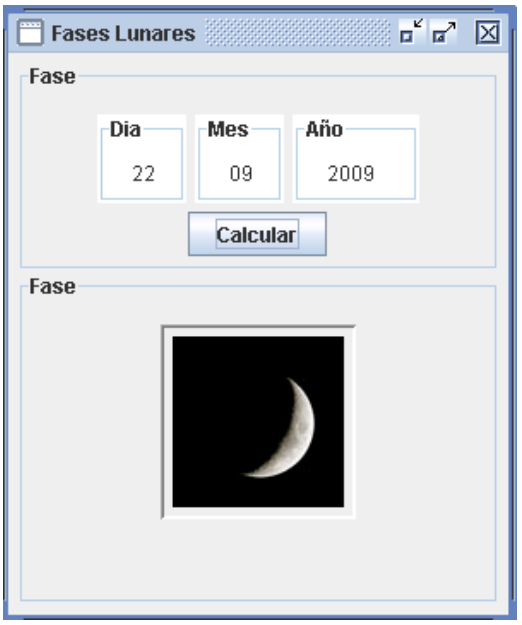
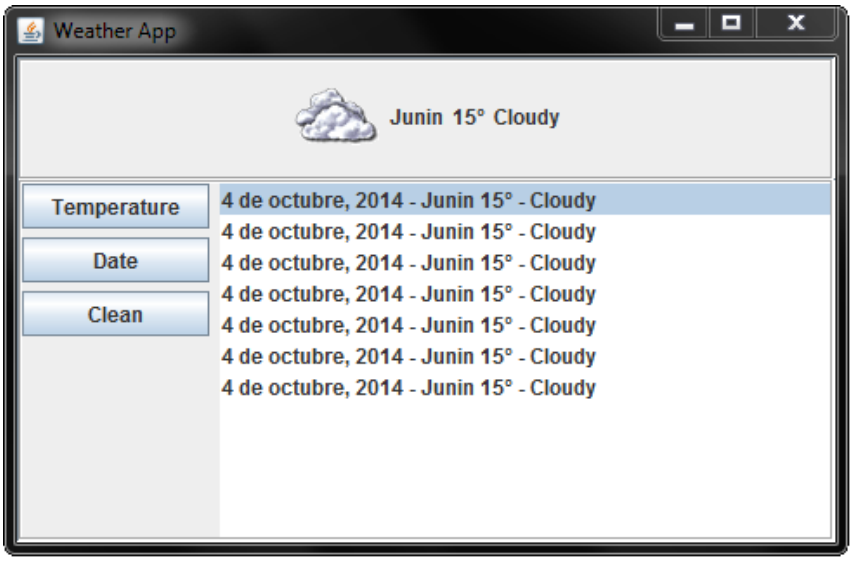


Figura 2

Ejercicio 3 – *Clima*

Realice la Interfaz de Usuario (IU) del ejercicio 6 del práctico 3. La interfaz de usuario debe ser similar a la que se muestra en el ejemplo. En la parte superior se muestra la temperatura actual. En la parte central se muestra el historial. Y en la izquierda se muestra una barra de botones con la siguiente funcionalidad:

- **Temperature**: ordena el historial por temperatura
- **Date**: ordena el historial por fecha
- **Clean**: borra el historial



Además, cuando se hace doble clic en un elemento de la lista se debe abrir una nueva ventana (diseño a elección) que muestre todos los datos del clima según el ejercicio 6 del práctico 1.

Nota: la cátedra provee (disponible en la *PlataformaEd*) una serie de objetos que funcionan como un servicio y proveen la información del clima. Su modelo debe interactuar con este servicio para poder obtener el clima.

Recuerde hacer la separación en capas usando MVC. Por un lado la capa de modelo/negocios que será la encargada de interactuar con el servicio del clima y de gestionar los climas y por otro lado la capa de presentación.

El modo de uso es el siguiente:

1. Para correr el servicio:

```
WeatherService  service = new WeatherService(City.Junin, 5);
service addObserver(this);
service.start();
```

2. Para parar el servicio:

```
service.stop();
service.deleteObserver(this);
```

3. Para recibir los updates:

```
@Override
public void update(Observable weather, Object param) {
	Channel channel = ((WeatherService)weather).getChannel();
	/*Channel posee toda la información necesaria*/
}
```

Diagrama de interacción de los servicios