Objetivos:

- Diseño OO, Uso de Colecciones
- División en capas MVC, Swing
- Event-Handling

En éste trabajo práctico vamos a utilizar Eclipse como IDE de desarrollo.

Ejercicio 1 – *Las Fases Lunares*

Se desea construir un programa (ver figura 1), 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, el cual es el encargado de realizar los cálculos y la GUI que es la encargada de mostrar los mismos.

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

```
/**
 * Calcula la posición de la luna
 */
private int determinarPosicion(Date d) {

        Calendar cal = Calendar.getInstance();
        cal.setTime(d);

        int dia = cal.get(Calendar.DATE);
        int mes = cal.get(Calendar.MONTH);
        int año = cal.get(Calendar.YEAR);

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

        if (dia < 3) {
                año--;
                mes += 12;
        }

        ++mes;

        c = 365.25 * año;
        e = 30.6 * mes;
        jd = c + e + dia - 694039.09;
        jd /= 29.5305882;
        b = (int) jd;
        jd -= b;
        b = (int)Math.round(jd * 8);
```

```
            if (b >= 8)
                b = 0;

            return b;
        }

        /**
         * Establece la Fase lunar
         */
        public Fase calcularFase(Date d) {

            switch (determinarPosicion(d) ) {
                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;
            }

throw new RuntimeException("Error al determinar la fase de la luna.");
                                                    }
```

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

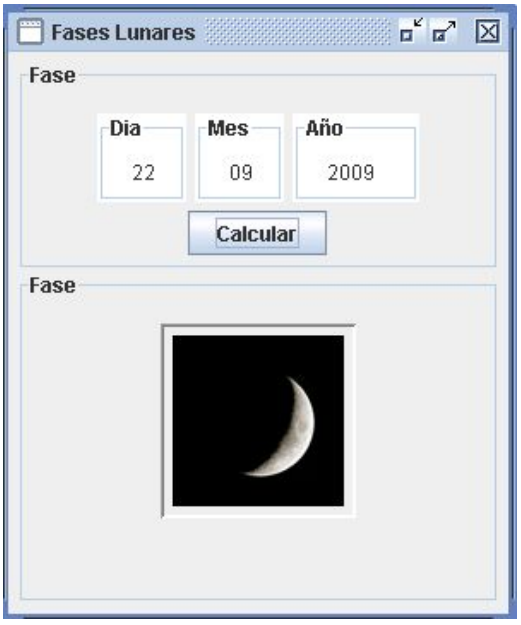


Figura 1

Ejercicio 2 – Conversor de Monedas

La agencia de cambio *Metropolis* 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 2 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 2

Ejercicio 3 – *Votación*

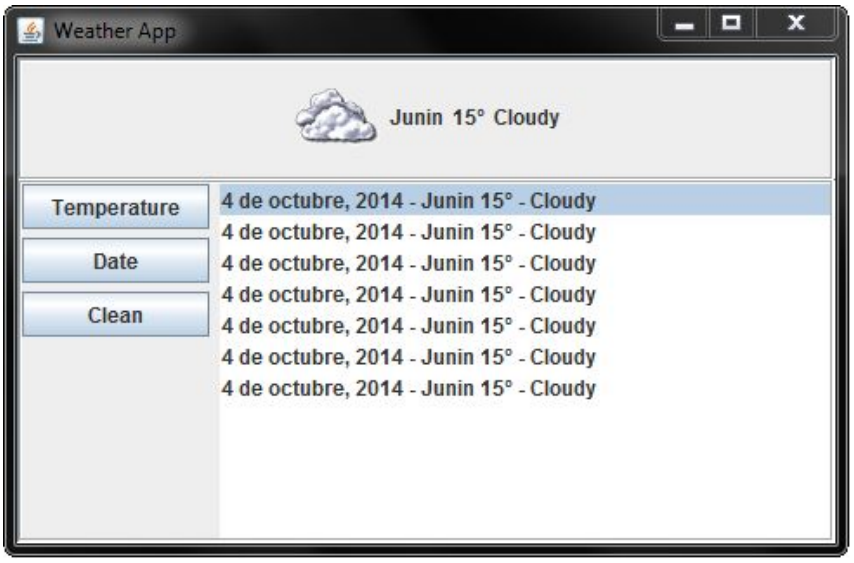
El Sindicato de Luz y Fuerza de la ciudad está solicitando la creación de un Sistema de Votación que le permita a sus afiliados poder elegir a sus candidatos para sus próximas elecciones. Este sistema tendrá un *Look and Feel* como la figura 3 y registrará la cantidad de votos de los candidatos a medida que los votantes expresen su elección. Es decir, cuando se presiona el botón Votar incrementara la cantidad de votos que tiene el candidato elegido.

Figura 3

Ejercicio 5 (ENTREGAR) – *Clima*

Realice la Interfaz de Usuario (UI) del ejercicio 6 del práctico 3. La interfaz debe ser similar a la que se muestra de 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 Moodle) 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. 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*/
}
```