Mensa 2: varias ventanas.

Cambiar los nombres de las variables asociadas a los objetos GUI que se utilicen en esta práctica.

Ventana Mensa:

1. Poner los componentes de la pestaña *Etiqueta*, son dos botones y un *jTextArea*.
2. Poner la imagen *checkIcon.png* al botón de la pestaña *Remitente*. Hay que mover el fichero al *package Mensa* y utilizar la propiedad *Icon* del botón.

Ventana VistaPrevia:

1. Crear la ventana *VistaPrevia* de manera que solo tenga un *jTextArea* que ocupe toda la misma.
2. Crear las estructuras de datos (Des, Ori y Texto) que mantienen la información sobre y la misma y los métodos set para modificarlas.

```
public VistaPrevia() {
        initComponents();
        Ori=""; Des=""; Texto="";
    }
private void repinta(){
        String TextoFinal="";
        if(Ori.length()>0)
            TextoFinal="Origen:"+Ori+"\n";
        if(Des.length()>0)
            TextoFinal=TextoFinal+"Destino:"+Des+"\n";
        TextoFinal=TextoFinal+Texto;
        this.taVistaPrevia.setText(TextoFinal);
    }
public void setDestinatario(String Des){
        this.Des=Des;
        repinta();
    }
public void setRemitente(String Ori){
        this.Ori=Ori;
        repinta();
    }
public void setTexto(String Texto){
        this.Texto=Texto;
        repinta();
    }
public void setColorFondo(java.awt.Color C){
        this.taVistaPrevia.setBackground(C);
    }
private String Des;
private String Ori;
private String Texto;
```

Ventana Opciones:

1. Crear la ventana *Opciones* como se muestra en el ejecutable.
2. Esta ventana necesita acceder a los componentes del nombre del remitente y destinatario de la ventana *Envio*, así como a los métodos *set* de la ventana *VistaPrevia*.
3. Utiliza el evento *ActionPerformed* por cada *checkBox* y *radioButton* de manera que se modifique la ventana de vista previa según las acciones como se muestra en el ejecutable.

Nota: creación del color azul claro:

```
AzulClaro=new java.awt.Color(200, 200, 255);
```

Ventana Inicial:

1. Crear la ventana *Inicial* como se muestra en el ejecutable.
2. Tiene que mostrarse antes que la ventana principal (*Envio*).
3. Tiene que responder ante eventos de tiempo (*Timer*) para que se desplace el *slider* y se actualice el texto como se muestra en el ejecutable. Cuando llegue al final se mostrará la ventana principal, ocultando ésta.
4. El botón cancelar tiene que terminar la aplicación(`System.exit(1);`).

Una posible implementación de esta clase es mediante un patrón *singleton* y generación de eventos *Timer*. Constructores:

```
private Inicial() {
    initComponents();

}

public Inicial(Envio VEnvio){
    this(); // Llamada al constructor void

    this.VEnvio=VEnvio;    // Guardar la referencia a la ventana princial
    this.setVisible(true); // Al ser ventana inicial es visible al crearse

    // Implementación del patrón singleton
    this.VInicial=this;

    // Generación de los eventos Timer
    java.util.Timer T=new java.util.Timer();
    // Programando 11 eventos
    for(int i=1;i<=11;i=i+1){
        java.util.TimerTask TTask=new java.util.TimerTask(){
                                    public void run(){Inicial.VInicial.manejadorTimer();}
                                  };

        T.schedule(TTask,i*250); // Cada 0.25s, tiempo en milisegundos, 1000ms=1segundo
    }
}
```