IVET CABRERA SANTANA

Proyecto reloj - alarma

```
public class ClockFunction extends JLabel implements Serializable{

    private boolean format24h;
    private Alarma alarmFunc;

    private SimpleDateFormat sdf24h;
    private SimpleDateFormat sdf12h;
    private Timer timer;
    private AlarmListener alarmListener;

    public ClockFunction(boolean on){

        this.format24h = on;

    }
```

Creo una clase serializable donde voy a crear todas las funciones para el reloj

```
public ClockFunction(){

    timer = new Timer(0, new ActionListener(){

        @Override
        public void actionPerformed(ActionEvent e){

            Date currentTime = new Date();
            if(format24h){

                String formatTime24 = "HH:mm:ss";
                sdf24h = new SimpleDateFormat(formatTime24);
                DateFormat formTime24 = new SimpleDateFormat(formatTime24);
                String formattedTime24 = formTime24.format(currentTime);
                setText(formattedTime24);

            }
            else{

                String formatTime12 = "hh:mm:ss a";
                sdf12h = new SimpleDateFormat(formatTime12);
                DateFormat formTime12 = new SimpleDateFormat(formatTime12);
                String formattedTime12 = formTime12.format(currentTime);
                setText(formattedTime12);

            }
```

En el controlador creo un reloj con los 2 formatos posibles. La mayor diferencia entre uno y otro son las HH mayúscula o minúscula. Hay formas más sencillas de hacerlo.

```
if(alarmFunc != null){

    if(alarmFunc.isOn() && timeMatch(currentTime, alarmFunc.getAlarmProgram())){

        if(alarmListener != null){

            alarmListener.rings();

        }

    }
```

Simplemente se le dice que si la alamra está activa y si coincide la hora actual con el reloj se active rings en el alarmListener

```
private boolean timeMatch(Date currentTime, Date alarmTime){

    Calendar calendar = Calendar.getInstance();
    calendar.setTime(currentTime);
    int actualHours, actualMin, actualSec, actualAlarmHours, actualAlarmMin, actualAlarmSec;
    actualHours = calendar.get(Calendar.HOUR_OF_DAY);
    actualMin = calendar.get(Calendar.MINUTE);
    actualSec = calendar.get(Calendar.SECOND);
    calendar.setTime(alarmTime);
    actualAlarmHours = calendar.get(Calendar.HOUR_OF_DAY);
    actualAlarmMin = calendar.get(Calendar.MINUTE);
    actualAlarmSec = calendar.get(Calendar.SECOND);

    if(actualHours == actualAlarmHours && actualMin == actualAlarmMin && actualSec == actualAlarmSec){
        return true;
    }else{
        return false;
    }

}
```

Método creado desde la foto anterior donde sigo los pasos del vídeo de ejemplo para configurar e igualar la hora actual con la hora de la alarma

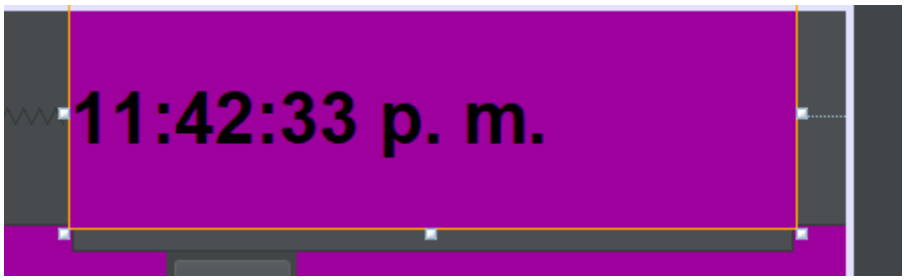
```java
public class Alarma implements Serializable{

    private Date alarmProgram;
    private boolean on;

    public Alarma(Date alarm, boolean active){

        this.alarmProgram = alarm;
        this.on = active;

    }

    public Date getAlarmProgram() {
        return alarmProgram;
    }

    public void setAlarmProgram(Date alarm) {
        this.alarmProgram = alarm;
    }

    public boolean isOn(){
        return on;
    }

    public void setOn(boolean active){
        this.on = active;
```

Se crea la clase alarma implementada serializable con los setters y getter de los métodos implantados por parámetro para saber si la alarma está activada y la hora de la alarma

```java
    public Alarma getSelectedValue(){

        Date date = (Date)jSpinner1.getValue();
        boolean on = jCheckBox1.isSelected();
        return new Alarma(date, on);

    }

    public void active(){

//        ClockFunction cf = new ClockFunction();
//
//        if (jCheckBox1.isSelected()) {
//
//            cf.setAlarmFunc(Date date, );
//
//        } else {
//            cf.setFormat24h(false);
//        }

    }
```

En el JPanel de la alarma comencé a hacer la función del botón de si la alarma está activada pero no me dio tiempo, por lo cual este botón no hace ninguna función

por lo cual para activar la alarma en esta prueba hay que ir al diseño de prueba del JFrame (Clock.java) y seleccionar el reloj



y configurarlo en el siguiente botón

En el JFrame llamo al ActionListener, el cual es una interfaz vacía y aquí le digo que el método creado dentro coja un archivo del sistema para que cuando se produzca el evento (alarma) suene la alarma que sirve con la librería importada.

```java
public class Clock extends javax.swing.JFrame {

    /**
     * Creates new form Clock
     */
    public Clock() {
        initComponents();
        this.setSize(600, 500);

        clockFunction1.addAlarmListener(new AlarmListener() {

            @Override
            public void rings() {

                try {
                    FileInputStream fis = null;
                    try {
                        fis = new FileInputStream("sounds/alarma.mp3");
                    } catch (FileNotFoundException ex) {
                    }
                    Player player = new Player(fis);
                    player.play();
                } catch (JavaLayerException ex) {
                }
            }

        });
```

```java
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    AlarmVisualPane avp = new AlarmVisualPane();
    avp.setBounds(0, 250, 600, 250);
    add(avp);
    avp.setVisible(true);

}
```

Al pulsar el botón de la alarma simplemente aparecerá debajo para configurar

Método creado para cambiar el formato cuando se le da l chekbox. El método se llama en el mismo checkbox

```
public static void format() {

    if (jCheckBox1.isSelected()) {

        clockFunction1.setFormat24h(true);

    } else {
        clockFunction1.setFormat24h(false);
    }
}
```