Certamen II ELO-329 Diseño y Programación Orientada a Objetos

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Segunda parte, **con apuntes**:

Segunda pregunta (35 pts.)

Se le ha encargado el desarrollo de la programación (no conducida por eventos) de un dispositivo el cual posee el comportamiento de un radio y un reloj. Para ello se le entrega el siguiente archivo disponible en la página de Aula:

- main.cpp el cual contiene la lógica de funcionamiento del dispositivo.

i. (10 pts.) Clase Radio, la cual define un dispositivo radio, teniendo el método switch_Radio() el cual define el encendido o apagado del mismo. Adicionalmente incluye el status de encendido o apagado del radio como atributo.

ii. (10 pts.) Clase Clock, la cual define un dispositivo reloj. Esta clase posee los atributos hour, minute y second, los cuales representarán la hora actual de un reloj. Adicionalmente, debe poseer un método tick() el cual hace que el reloj mueva (incremente) su hora actual en 1 segundo. Para incrementar el tiempo se debe tener en cuenta que existe un orden para incrementar, por ejemplo, primero los segundos, después los minutos y por último las horas.

iii. (15 pts.) Clase RadioClock con alarma, la cual hereda de la clase Radio y la clase Clock define un dispositivo compuesto por un radio y un reloj. Esta clase posee los atributos a_hour, a_minute y a_second los cuales representan la hora en que la alarma del dispositivo debe activarse. Adicionalmente, un atributo status establece si esta alarma está activada o no. Además, se cuenta con los siguientes métodos:

    - void setAlarm(int pHour, int pMinute, int pSecond), el cual establece la hora de sonido de la alarma en el dispositivo.
    - void switch_Alarm(), el cual establece el estado de apagada o encendida de la alarma en el dispositivo.
    - bool verify_Alarm(), el cual verifica si la alarma debe activarse o no. Esto debe suceder cuando la hora actual y hora de alarma coinciden.
    - void activate_Alarm(): el cual muestra un mensaje de “Beep” si es que la alarma está verificada para activarse. Para mostrar este mensaje se debe considera que el radio tenga el status de encendido activado, en caso contrario activar.

Respuesta: Usted puede bajar el Código desde
http://profesores.elo.utfsm.cl/~agv/elo329/1s23/ELO329_1s23_C2P2_Sol/

//main.cpp

```
#include <iostream>
#include "RadioClock.h"
using namespace std;
int main()
{
int hour = 11, minute = 25, second = 15;
int a_hour = 13, a_minute = 26, a_second = 16;

RadioClock radioAlarma(hour, minute, second, true, true);
radioAlarma.setAlarm(a_hour, a_minute, a_second);
while (!radioAlarma.verify_Alarm()) {
radioAlarma.tick();
radioAlarma.showTime();
}
return 0;
}
```

i) **//radio.h**

```
#include <iostream>
//Definición correcta de clase 1pt
class Radio
{
protected:
        //Definición correcta de atributo 1pt
bool power;
public:
        //Definición correcta de constructor 1pt
Radio(bool pPower);
        //Definición correcta de método 1pt
void switch_Radio();
};
```

//radio.cpp

```
#include "Radio.h"
//Implementación correcta de constructor 3pt
Radio::Radio(bool pPower)
{
```

```
power = pPower;
}
 // Implementación correcta de método 3pt
void Radio::switch_Radio()
{
power = !power;
}
```

ii) **//clock.h**

```
#include<iostream>
using namespace std;
//Definición correcta de clase 1pt
class Clock
{
protected:
        //Definición correcta de atributos 1pt
int hour;
int minute;
int second;
public:
        //Definición correcta de constructor 1pt
Clock(int pHour, int pMinute, int pSecond);
        //Definición correcta de método 1pt
void tick();
        //Definición correcta de método 1pt
void showTime();
};
```

//clock.cpp

```
#include "Clock.h"
//Implementación correcta de constructor 1pt
Clock::Clock(int pHour, int pMinute, int pSecond) {
hour = pHour;
minute = pMinute;
second = pSecond;
}
//Implementación correcta de método 3pts
void Clock::tick()
{
if (second < 59) {
second++;
}
else if (minute < 59)
```

```
{
minute++;
second = 0;
}
else if (hour < 23)
{
hour++;
minute = 0;
second = 0;
}
else
        {
hour = 0;
minute = 0;
second = 0;
}
}
//Implementación correcta de método 1pt
void Clock::showTime() {
cout << endl << hour << " " << minute <<" " << second << endl <<endl;
}
```

iii) **//radioclock.h**

```
#include "Radio.h"
#include "Clock.h"
#include <iostream>
using namespace std;

//Definición correcta de clase 2pts
class RadioClock : public Radio, public Clock
{
private:
        //Definición correcta de atributos 1pt
int a_Hour;
int a_Minute;
int a_Second;
bool status;
void activate_Alarm();
public:
        //Definición correcta de constructor 1pt
RadioClock(int pHour, int pMinute, int pSecond, bool pPower, bool pStatus);
        //Definición correcta de método 1pt
void setAlarm(int pHour, int pMinute, int pSecond);
```

```
        //Definición correcta de método 1pt
        void switch_Alarm();
        //Definición correcta de método 1pt
        bool verify_Alarm();
};
```

//radioclock.cpp

```
#include "RadioClock.h"
//Implementación correcta de constructor 1pt
RadioClock::RadioClock(int pHour, int pMinute, int pSecond, bool pPower, bool pStatus) : Radio(pPower), Clock(pHour, pMinute, pSecond)
{
status = pStatus;
}
//Implementación correcta de constructor 2pt
void RadioClock::setAlarm(int pHour, int pMinute, int pSecond)
{
a_Hour = pHour;
a_Minute = pMinute;
a_Second = pSecond;
}
//Implementación correcta de método 1pt
void RadioClock::switch_Alarm()
{
status = !status;
}
//Implementación correcta de método 2pt
bool RadioClock::verify_Alarm()
{
if (hour == a_Hour && minute == a_Minute && second == a_Second) {
activate_Alarm();
switch_Alarm();
return true;
}
return false;
}
//Implementación correcta de método 2pt
void RadioClock::activate_Alarm()
{
if (power==false)
switch_Radio();

for (int i = 0; i < 20; i++)
cout << "Beep!" <<endl;
```

```
}
```