

## PROGRAMACION

NOME:.....

A nosa empresa decidiu implantar un sistema de control presencial, e para elo mercou 200 aparellos de control presencial "TecnoWorkControl" basados en Java. Son aparellos moi sinxelos e de baixo custo (20€ a unidade) que simplemente dispoñen dun frontal cun lector de tarxetas, e dous botóns (**In** e **Out**) e que están controlados mediante unha aplicación Java que non suministra o fabricante, debendo ser creada pola empresa segundo as súas necesidades implementado a interface **TecnoWorkControlDriver**.

O programa encargado de interactuar coa máquina xa ven instalado no sistema e proporciona unha clase Java pertencente ao package **TecnoWorkDriver** que implementa o seguinte interfaz:

```
public interface TecnoWorkControlManager {  
    /** ledOn  
        Prende o LED do sistema na cor indicada (1-Blue, 2-Red, 3-Yellow, 4-Green, 5- RedBlink)    */  
    public void ledOn(int color) ;  
  
    /** ledOff  
        Apaga o LED do sistema    */  
    public void ledOff();  
  
    /** passive  
        Pon o sistema en modo 'pasivo', no que o lector de tarxetas non é funcional.    */  
    public void passive();  
  
    /** active  
        Pon o sistema en modo 'activo', no que o lector de tarxetas lee. A os 20 segundos sin actividade se pon  
        automáticamente en pasivo    */  
    public void active();  
  
    /** openDoor  
        Abre a porta á que está asociada o aparello    */  
    public void openDoor();  
}
```

**Obxecto de Control do Dispositivo facilitado polo fabricante**



**TecnoWorkControlManager**

ledOn  
ledOff  
passive  
active  
openDoor

**Aplicación da Empresa**

**TecnoWorkControlDriver**

No arranque o dispositivo chama a Manager  
initPresenceControl pasándolle o obxecto de control

**Manager**

```
static TecnoWorkControlDriver initPresenceControl(TecnoWorkControlManager mgr);
```

O **Manager** lle devolve un obxecto de este tipo ao dispositivo no arranque cando invoca a initPresenceControl

## O funcionamento do sistema é o seguinte:

O sistema se instala nas portas de acceso á empresa controlando a apertura das mesmas. A empresa debe insertar en cada sistema unha tarxeta micro-sd coa aplicación Java de control que desenvolva. A clase principal que implementa o interfaz do driver (*TecnoWorkControlDriver*) debe estar no package **TecnoWorkDriver** (as clases auxiliares non teñen por qué, e a empresa decidiu que irán no package **ControlPresencial**).

Cando o aparello arranca, ao final do seu proceso de inicialización, chama ao método estático da clase Manager *initPresenceControl* pasándolle como argumento o obxecto *TecnoWorkControlManager* que ven instalado de fábrica, e obtendo como resposta un obxecto **TecnoWorkControlDriver** que o dispositivo utilizará para interactuar coa aplicación. **Esta clase Manager é parte da aplicación que debe desenvolver a empresa.**

Unha posible implementación deste comportamento sería o seguinte (NOTA: **Sería parte do código da aplicación de control que ven instalada de fábrica**):

```
(Proceso previo de arranque do dispositivo, como a creación do obxecto interno
TecnoWorkControlManager)
[.....]
TecnoWorkControlDriver
_DRV=Manager.initPresenceControl(obxecto_interno_TecnoWorkControlManager);
[...]
```

A partir de aquí, o aparello xa pode interactuar co driver da empresa mediante o obxecto **\_DRV** do seguinte xeito:

- Cando un usuario preme o botón "**In**", a máquina chamará ao método **inButtonPressed** de **\_DRV**.
- Cando un usuario preme o botón "**Out**", a máquina chamará ao método **outButtonPressed** de **\_DRV**.
- Cando un usuario pasa unha tarxeta polo lector, e **si o lector está en modo "active"**, a máquina chamará ao método **identificationRead** de **\_DRV** pasándolle como argumento o **ID** único do usuario leído da tarxeta que é de tipo **String**.

**NOTA:** Fixádevos que cando o aparello chama a *initPresenceControl* lle pasa un obxecto *TecnoWorkControlManager*. A aplicación da empresa deberá utilizar este obxecto para implementar o seu control (prender os leds, apagalos, ler tarxetas....)

---

## EXERCICIO 1

Definir a interface **TecnoWorkControlDriver**

## EXERCICIO 2

Crear a Aplicación **PresenceControl** cunha clase implementando **TecnoWorkControlDriver** chamada **MyEnterprisePresenceControl** co seguinte comportamento:

- Cando o usuario preme o botón "**In**" no aparello, se prenderá o led en cor azul, e se porá o aparello en modo lectura (activo).
- Cando o usuario preme o botón "**Out**" no aparello, se prenderá o led en cor verde, e se porá o aparello en modo lectura (activo).
- Cando se lea unha tarxeta, de momento non fará nada mais que poñer o led en vermello parpadeante, o aparello en modo pasivo, abrir a porta e apagar o led.

### EXERCICIO 3

A empresa dispón de 3 tipos de Persoal: **Xefes, Currantes e Accionistas**. Se desexa levar o seu control de presenza instalando en todas as entradas da empresa aparellos deste tipo e rexistrando os accesos nunha base de datos, non permitindo accesos non autorizados. A estrutura da BBDD é a seguinte:

<b><u>Persoal:</u></b> ID : String (corresponde aos ID das tarxetas) Nome: String Telefono: String E-Mail: String Tipo: int (0 Traballador, 1 Xefe, 2 Accionista)	<b><u>Presencia:</u></b> ID: String (ID do traballador) Entrada: long (Calendar.getInstance().getTimeInMillis()) Saida: long (Calendar.getInstance().getTimeInMillis()) Bono: int (Penalización/Bonificación)
--	---

Se desexa que cando un *Traballador* entre máis tarde das 10h:00m reciba un "bono" de penalización de -5€, e que cando salga despois das 19h:00h un "bono" de 5€. Os *Xefes* e *Accionistas* non teñen penalizacións nin bonos, pero os Accionistas non poderán entrar na empresa despois das 14h:00m.

Como se desexa almacenar os datos na "Nube", se encarga á empresa "NubeSolutions Inc." a programación dun compoñente (clase) Java que implemente o seguinte interfaz:

```
public interfaz KeepPresence {  
    /**  
        registerPresence  
        Rexistra a información de presenza na BBDD. O obxecto persoal xa debe levar no seu atributo  
'bono' a bonificación/penalización a aplicar.  
        Os Traballadores, como os Xefes e os Accionistas son Persoal. type pode ser: 0 - Entrada, e 1 -  
        Saída.  
    */  
    public void registerPresence (Personal p, long time, int type) throws Exception;  
  
    /**  
        getPersonal  
        Devolve o Personal identificado por ID, lanzando unha Exception si non existe na BBDD.  
        O Personal devolto será un Traballador, Xefe, ou Accionista segundo o valor do seu atributo Tipo.  
    */  
    public Personal getPersonal(String ID) throws Exception;  
}
```

O compoñente mercado chámase **DropBoxPresence.class**. e almacena a información de presenza nunha BBDD en DropBox.

**SE PIDE** modificar a aplicación desenvolta no punto 2 (PresenceControl) de xeito que rexistre a información nunha BBDD aloxada en DropBox mediante a clase **DropBoxPresence.class**.

### EXERCICIO 4

A empresa chegou á conclusión, despois de meses de estudio, que almacenar este tipo de información en *DropBox* non é unha boa idea, polo que decide reimplementar a aplicación para que faga uso dunha BBDD MySQL. **SE PIDE** modificar a aplicación PresenceControl de xeito que a información se rexistre nunha BBDD MySQL chamada PresenceDatabase mediante JDBC.

#### AVALIACION:

- 1.- 1 pts (Se valora a capacidade identificación e definición de interfaces)
- 2.- 3 pts (Se valora a comprensión do uso das interfaces e a súa utilidade)
- 3.- 3 pts (Se valora o emprego da Herencia, do Polimorfismo e da Sobreposición e a comprensión do desenvolvemento compartido de aplicacións)
- 4.- 3 pts (Se valora a comprensión do desenvolvemento modular e o uso de bases de datos mediante JDBC)