Programación.	Prácticas obligatorias

Normativa:

- Las siguientes prácticas **son obligatorias** para aprobar la asignatura y tienen que **ser entregadas y explicadas antes de la fecha fijada** por el profesor.
- El material a entregar será un **diagrama UML por cada clase** que se diseñe y **el código fuente de la práctica en el lenguaje requerido.**
- **No se corregirán prácticas que no compilen.**
- Las prácticas **son individuales**.

Práctica 1.Java

Un programa de dibujo simple trabaja con una serie de tipos de figuras geométricas como son el *Circulo* y el *Cuadrado* .Se debe poner calcular sobre cuqluier tipo de *Figura* el valor del área

Se pide:

a) Diagrama de clases UML
b) Una **Clase ProgramaPrincipal que defina** el siguiente método prototipo :

```
private void calcularArea(…..){
//obtener el área de la figura e imprimirla por pantalla
}
```

c) En el método **main()** realice lo siguiente:
   - Crear un cuadrado lado=4
   - Crear un circulo con radio=6
   - obtener el área del circulo y cualdrado através del método **calcularArea(..) mencionado anteriormente.**

Práctica 2. Java

Un módulo de control de tareas ejecuta diversos tipos de tareas. Una ***Tarea*** se define a partir de **un** script o guion (cadena) que indica las instrucciones que hay que ejecutar, un valor de prioridad (valor 1..3) y un GUID (numérico). El módulo trabaja fundamentalmente con dos tipos de Tareas: ***Tarea Planificada y Tarea Normal.*** Las características de cada tipo de tarea son las siguientes.

- **Tarea Normal:** Cuando el modulo de control la ejecute (llamada a ejecutar ()) **, s**e ejecutará siempre que se cumpla la condición GUID=0 y prioridad > 2 dentro de la tarea.
- **Tarea Planificada:** Es una tarea que tiene una fecha de ejecución (día (numérico) y mes (numérico)), y cuando el modulo de control la ejecute sólo se ejecutará si su fecha interna (previamente establecida) coincide con la fecha del instante de ejecución. Si se intenta ejecutar la tarea fuera de esa fecha sencillamente no pasará nada.

  Un ejemplo de la tarea de planificación podría ser día=1 mes=12 lo que indicaría que la tarea sólo se ejecutaría dentro del módulo si ocurre el 1 de diciembre.

- Todas las tareas tienen además un indicador de estado(0/1) que indica si la tarea ya ha sido ejecutada estado=1 o no luego estado=0.

La ***clase modulo de control de tareas*** dispone de dos métodos: ***registrar(Tarea)*** y ***ejecutarTareas().*** El primer método registra una tarea dentro del módulo para ser ejecutada posteriormente y el segundo método ejecuta todas y cada una de las tereas registradas hasta ese momento, además cambia el estado interno de cada tarea (1 si se ejecuto o 0 sino).

Se pide:

- Modele la relación de generalización /especialización de las tareas del enunciado así como la asociación de estas con la clase módulo de control.
- Cree una clase **ProgramaPrincipal** que
  - Cree un **array de tamaño 3**
  - Cree 1 Tarea Ordinaria (T1) y 2 Tareas Planificadas (T2,T3)
  - Establezca la fecha en la que tiene que ejecutar la T2 (Ejemplo fecha de hoy) y T3 (día tres del mes que viene).
  - Registre las tareas en el módulo de control de tareas
  - Ejecute todas las tareas y compruebe el estado de cada tarea. Muestre por pantalla este estado

Consideraciones sobre la práctica:

- Para obtener el día y el mes se debe usar la clase **GregorianCalendar** (**import java.util.Calendar y import java.util.GregorianCalendar**) . El siguiente fragmento de código permite obtener el día y mes actual. Utilice este código para la **Tarea Planificada**

```
GregorianCalendar fechaActual=new GregorianCalendar();//crea un objeto representando la fecha actual

int diaMes=fechaActual.get(Calendar.DAY_OF_MONTH);//Obtiene el día del mes actual

int Mes=fechaActual.get(Calendar.MONTH);//Obtiene el mes del año actual
```

- El atributo script es de tipo cadena y tiene el siguiente formato:

  "I1:comando1,I2: comando2,…".

- Se considera que el comportamiento del método ejecutar de una tarea es el siguiente
  - Prototipo String ejecutar()
    - devuelve la cadena **"EJECUCIÓN : script ={ I1:comando1,I2:comando2}"** si la ejecución puede realizarse o la cadena **" NO ES POSIBLE LA EJECUCIÓN"** en otro caso.

Práctica 3.C++

Un sistema Domótico usa los siguientes tipos de sensores entre otros: *Humedad y Temperatura*. En general un **Sensor** se caracteriza por las siguientes propiedades:

- Un identificador de dispositivo**(cadena "ID00N")**
- Lectura **(numérico)**
- Unidades de interpretación de la lectura (HumendadRelativa % ,Temperatura(Celsius) **(cadena)**
- Estado actual **(cadena)**

De manera general un **Sensor** se puede **activar, leer.** Los estados posibles para cada tipo de sensor se muestran en la tabla siguiente:

Tipo de sensor	Estado (inicial)	Estados posibles
Sensor Humedad	"Operativo"	"Error","Operativo"
Sensor Temperatura	"Operativo"	"Error","Operativo","Perdida Contacto"

Cada sensor implementa la operación ***activar()*** cambia el estado del sensor en de la siguiente forma:

- Sensor Humedad: Estado ="Operativo"
- Sensor Temperatura: Estado ="Operativo"

La operación ***double leer()*** de cada sensor tiene el siguiente comportamiento

- Sensor de Humedad y Sensor Temperatura: Si el sensor está en estado "Operativo" devuelve la **última lectura obtenida** en otro caso -1.

Además se dispone de una clase denominada **Detector** que permite obtener el **vector de estado** a través de un método denominado **obtenerEstado()**, de todos los sensores registrados(método registrar(Sensor)) .la clase Detector además implementa un método denominado **activarSensores()** que activa todos y cada uno de los sensores registrado.

Se pide :

- Cree un modelo UML de las clases implicadas en el ejemplo
- Cree una clase **ProgramaPrincipal** que en su método main realice lo siguiente:
    - instancie 2 objetos de tipo Sensor, uno de cada tipo.
    - Instanciar una objeto **Detector**
    - Registrar los dos sensores en el ámbito del módulo **Detector** a través del método ***registrar (Sensor)***.
    - **Y realizar la siguiente operación con el objeto Detector :**
        - Activar todos los sensores.
        - Obtener el vector de estado

Como ejemplo considere la siguiente salida en el programa principal

<**ID001**:Estado:"Operativo":Lectura:13.4,

ID002:Estado:"Error":Lectura:_,

ID003:Estado:"Operativo":Lectura:30:" ,….>