Universidad de Jaén
Departamento de Informática

Programación Orientada a Objetos

https://tiny.cc/poouja

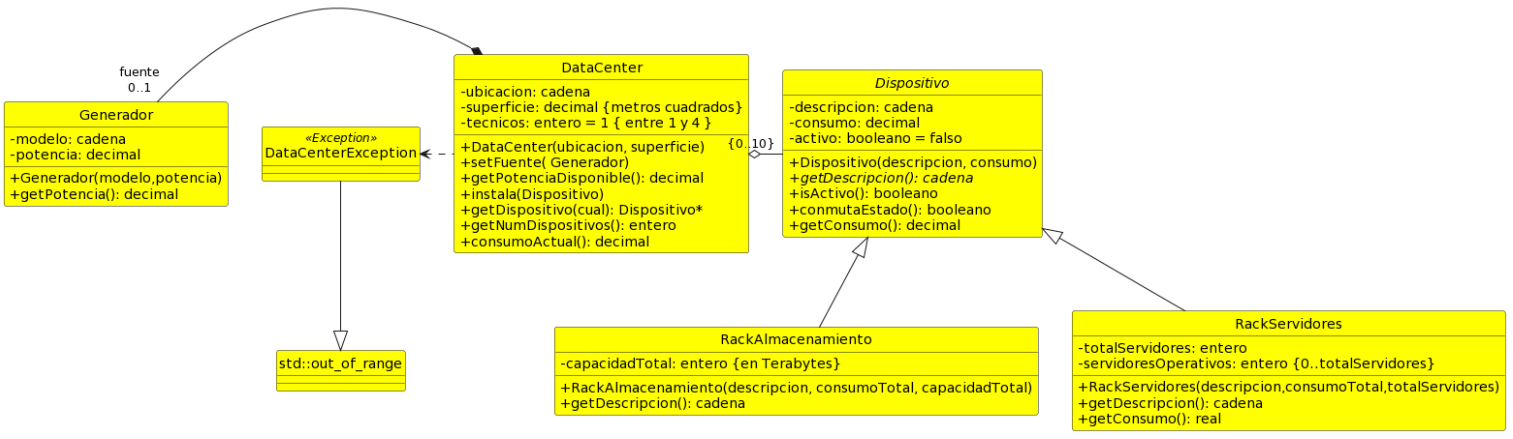
SEMINARIO DE PREPARACIÓN DE LA PRUEBA PRÁCTICA 2

Instrucciones

- Rellene la parte superior de esta hoja indicando su nombre, apellidos y el puesto que ocupa.
- **Dispone de 1 hora y 40 minutos** para la realización de los ejercicios propuestos.
- **Material:** Sólo puede hacer uso para la realización de la actividad del material de teoría y guiones de prácticas accesible en el espacio virtual de la asignatura. También dispone de código fuente de algunas partes del problema que debe completar siguiendo los ejercicios propuestos. El profesor le aclarará la sintaxis de cualquier duda que tenga sobre cualquier función o clase de la biblioteca estándar de C++ que necesite para la implementación siempre y cuando este considere que es necesaria para su realización. Para la edición del código, compilación y depuración debe utilizarse el ordenador del aula con cualquiera de los entornos de desarrollo y compiladores instalados.
- **Recomendaciones**: grabe continuamente el código generado en disco para evitar pérdidas de información en caso de bloqueo de la máquina. Si esto ocurriera, notifíquelo al profesor para que le indique cómo proceder. Vaya implementando, compilando y probando cada ejercicio de forma individual. Si no sabe cómo implementar algún método, implemente al menos su cabecera y deje el contenido en blanco o comentado. Al menos podrá llamarlo desde donde proceda y continuar realizando el resto de ejercicios. Si encuentra errores de compilación que no es capaz de resolver, comente el fragmento de código que genere los problemas y siga trabajando con el resto de ejercicios.
- **Está prohibido expresamente** utilizar cualquier otro material impreso, en formato digital o accesible de forma on-line para la realización del ejercicio. Tampoco se pueden realizar copias del presente enunciado o del material realizado en cualquier formato ni sacarlas del aula utilizando cualquier soporte de almacenamiento o servicio telemático. Cualquier incumplimiento de estas normas supondrá el suspenso inmediato del ejercicio.
- Al finalizar la actividad, **entregue todos los ficheros de código fuente y el último ejecutable obtenido**, comprimidos en un fichero en **formato .ZIP** a través del espacio virtual de su grupo de prácticas, en la actividad creada a tal efecto. **Entregue también esta hoja** debidamente cumplimentada.
- *No olvide borrar del ordenador todo el código realizado al finalizar su trabajo* y después de asegurarse de que lo ha enviado correctamente. Si tiene dudas al respecto, pregúntele al profesor si ha realizado correctamente la entrega.
- *Podrá acceder a todo el material que entregue así como a este enunciado a partir del día siguiente a la realización de esta actividad* en el espacio virtual de su grupo de prácticas.

Descripción de la actividad

Con los ingresos obtenidos en el primer contrato firmado con una empresa de Laponia, *CloudPOO* decide ampliar la funcionalidad de su solución de monitorización añadiendo características específicas a los diferentes tipos de dispositivos que se pueden monitorizar. Adaptar el código C++ suministrado a partir del siguiente diseño UML y las funcionalidades descritas en los ejercicios. Para comprobar su correcto funcionamiento, **realizar las pruebas descritas en la función *main***.



Ejercicios

NOTA: si el proyecto no compila la calificación se reducirá un 10%

1.- (2 puntos) Crear las **clases *RackServidores y RackAlmacenamiento*** en sus módulos (*no in-line*) con atributos y visibilidad indicados. Implementar los constructores parametrizados indicados y métodos get/set para atributos específicos. Si en un *RackServidores* se intentan marcar como operativos más servidores de los disponibles se debe lanzar alguna excepción adecuada de la jerarquía STL.

2.- (1,5 puntos) Adaptar la clase *Dispositivo* para que puedan redefinirse los siguientes métodos e implementarlos, en su caso, en sus derivadas: ***getDescripcion()*,** debe redefinirse obligatoriamente en cualquier derivada para que, además de la descripción, incluya entre paréntesis su característica principal, e.g. número de servidores operativos o capacidad total de almacenamiento; ***getConsumo(),*** en un *RackServidores* será proporcional al número de servidores operativos que queden, e.g. si el consumo máximo es de 1000W para un rack con 10 servidores, si solo quedaran operativos 6, el consumo real sería solo de 600W

> **Prueba 1 (0,5 puntos)**. Crear dos Racks de servidores: el primero con 1000W de consumo máximo y 10 servidores en total, el segundo con 2000W y 20 servidores. Crear dos Racks de almacenamiento con consumos de 800W y 1500W, y capacidades de 10 y 100 Terabytes respectivamente. Reducir el número de servidores operativos del primer rack a 9 y visualizar su descripción y consumo. Capturar y mostrar cualquier excepción que pudiera ocurrir durante las operaciones realizadas.

3.- (1,5 puntos) Operadores de asignación de las clases de la jerarquía de dispositivos

4.- (1 punto) Crear la clase **excepción *DataCenterException*** según diseño. **Adapta el método *instala()* de *DataCenter*** para que además lance esta nueva excepción si la potencia del dispositivo a instalar supera a la potencia disponible del DataCenter .

> **Prueba 2 (0,5 puntos)** Asignar al segundo rack de servidores creado el primero de ellos. Instalar en el DataCenter el segundo rack de servidores y los dos racks de almacenamiento. Capturar, usando polimorfismo, cualquier excepción que pudiera generarse en el proceso. Si la excepción capturada es de tipo *DataCenterException*, además del motivo debe mostrarse la potencia disponible del DataCenter en ese momento.

5.- (2 puntos) Implementar el método ***Dispositivo* DataCenter::mayorConsumo()*** que devuelve el dispositivo con mayor consumo del DataCenter.

6.- (0,5 puntos) Modifica la **función visualiza** en main para que muestre todos los detalles de un dispositivo dependiendo del tipo al que pertenece.

> **Prueba 3 (0,5 puntos)** En main, localizar el dispositivo con mayor consumo del DataCenter y, si existe, visualiza sus datos con la función del ejercicio anterior