TAREA PARA PROG08

Detalles de la tarea de esta unidad.

Enunciado.

Es imprescindible para realizar la práctica, trabajar bien el anexo que está en el aula. La mayoría de los conceptos que se necesitan para esta práctica están en ese anexo, además de trabajar los contenidos oficiales que los amplían.

Consideraciones sobre el código que implementes:

** La práctica se tiene que hacer partiendo de cero. Si se detectan copias parciales de código fuente otras prácticas, se dará por nula.*

** El código tiene que estar suficientemente comentado y bien indentado (también conocido como sangrado), si no, no se corregirá por ilegibilidad del mismo.*

*Como ya sabrás qué es de javadoc, si escribes /** y pulsas intro, automáticamente se crea una zona de comentarios en las que aparece el autor, y podrías añadir más datos, como la versión, fecha o lo que veas oportuno. Es conveniente hacerlo antes de cada clase, e incluso de cada método.*

**Además, hay que poner un comentario al principio cada clase, con tu nombre y apellidos, indicando el tema en el que estamos. Todo esto es requisito para continuar la corrección.*

PARTE 1

Se quiere realizar una aplicación que simule algunas operativas sobre Ordenadores.

Para ello se crearán varias clases en el paquete:

maestre.ordenadores.modelo

Todas las clases tendrán sus atributos encapsulados (get y set, y visibilidad adecuada de los atributos), y **2 constructores**. Uno con valores por defecto y otro recibiendo todos los parámetros que inicializarán los atributos.

Además, todas tendrán un **toString**(), sobreescribiendo el de la clase Object, devolviendo una cadena con los datos oportunos.

Además, la clase ordenador, sobrescribirá el **equals**, de la clase Object. 2 ordenadores, son iguales, si el número de serie es igual.

De todos los ordenadores se necesita saber: (Clase **Ordenador**)

- **Número de serie**: Será un texto formado por letras y números de máximo 20 caracteres.
- **Marca y modelo**: texto.
- **Memoria RAM**: un número entero de como máximo 3 dígitos, que serían los GB. (elige el tamaño apropiado)
- **Procesador**: texto.
- **Núcleos**: un número entero de como máximo 2 dígitos, que serían el nº de núcleos.
- **Tipo disco duro**: HDD / SDD
- **Arrancado**: Indica si el ordenador está en marcha, es decir, encendido o no.

Las operaciones que se pueden realizar sobre un ordenador son:

- **Arrancar**: Indicará un mensaje por pantalla que el ordenador se está arrancando, luego ya se dirá que se ha arrancado, actualizando además el estado del ordenador (arrancado, pasa a true). Puedes dotarle de símbolos o algún mensaje propio que quede “chulo” y simule un arrancado.
- **Apagar**: Idem, pero apagando y dejando el estado Arrancado en falso.

- **Dibujar**: Será un método abstracto que deberán implementar las hijas, lo que hace que la clase Ordenadores, sea abstracta, es decir no se pueden crear objetos de ella, sólo de sus descendientes.

Además de esas características que tendrán todos los ordenadores, estos se dividen en 2. Portátiles y Sobremesa.

De los **portátiles**, además se quiere guardar: (Clase Portatil)

- **Pulgadas** de la pantalla: número entero de 2 dígitos.
- **Duración Batería**: un número que indica los minutos de duración aproximada que le quedan a la batería (empieza con 300 minutos).
- **Cargar** (se suma a la batería los minutos que se indiquen, pasando como parámetro)
- **Descargar** (se resta a la batería los minutos que se indiquen, pasando como parámetro)
- **Dibujar** (si lo estás haciendo bien, este método es obligatorio ya que en la clase Ordenador lo pusiste como abstracto)
    - Muestra el ordenador de la siguiente manera:
        - Indica los Datos del ordenador (usa el toString(), incluido datos de la clase ordenador y los propios de portátil. Además, escribirá:
        - **Una línea con asteriscos por cada núcleo**.
        - Un ***** por cada **GB** de memoria RAM (dentro de la línea en la que esté) que tiene siempre que el ordenador esté arrancado. Si no está arrancado se indicará un mensaje diciéndolo.
        - El pintado se realizará **con estructuras repetitivas** según los valores indicados anteriormente.

Ejemplo: Para un portátil de 8 GB y 4 núcleos: (4 líneas de 8 asteriscos cada una)

Además de los datos del ordenador, se muestra lo siguiente:

```
********
********
********
********
```

De **los ordenadores de sobremesa**, además se quiere guardar: (Clase Sobremesa)

- **Placa Base**: texto.
- **Tarjeta gráfica**: texto.
- **Dibujar** (si lo estás haciendo bien, este método es obligatorio ya que en la clase Ordenador lo pusiste como abstracto.
    - Muestra el ordenador de la siguiente manera:
        - Indica Datos del ordenador, incluido datos de la clase ordenador y los propios de portátil. Además, escribirá:
        - **Una línea con asteriscos por cada núcleo**.
        - Un ***** por cada **GB** de memoria RAM (dentro de la línea en la que esté) que tiene siempre que el ordenador esté arrancado. Si no está arrancado se indicará un mensaje diciéndolo.
        - El pintado se realizará **con estructuras repetitivas** según los valores indicados anteriormente.

    Como en ejemplo anterior, pero se diferencian los datos del ordenador, por eso es necesario implementarlo en cada subclase, y no en la clase padre.

Con todo ello, se pide:

1. Del texto anterior se deben **extraer** las **clases** y **relaciones de herencia** necesarias.
2. La aplicación deberá llevar un correcto control de **excepciones** (genéricas y de las clases que crees si son necesarias)
3. Utiliza **super** y **this** cuándo lo veas oportuno.
4. Documenta con nomenclatura javadoc.
5. Se realizará una aplicación, llamada **Parte1app**, tipo consola en el paquete **maestre.ordenadores.app**

6. La aplicación tendrá un **array** de **Ordenadores** precargado, es decir tú dejas en código relleno esos datos (3 portátiles y 3 sobremesa, todos en el mismo array, ya que en prácticas anteriores hemos pedido muchas veces datos por pantalla y hemos ido almacenando, en esta me vale con que esté precargado)
7. Se inicia dando un **mensaje de bienvenida** y recorriendo el array, mostrando los datos de cada ordenador.
8. Después aparecerá un **menú** en el que se podrá:
    a. Listar todos los ordenadores mezclados portátiles y sobremesa.
    b. Listar sólo los portátiles. (¿instanceof?)
    c. Listar sólo los ordenadores de sobremesa. (¿instanceof?)
    d. Encender un ordenador (se pide la posición)
    e. Apagar un ordenador (se pide la posición)
    f. Dibujar un ordenador (se pide la posición)
    g. Cargar Portátiles. (pedirá un número que serán los minutos que recargaremos todos los portátiles del array (necesitas insanceof, ya que los ordenadores de sobremesa no tienen batería)
    h. Descargar Portátiles. (igual, pero descargando).

PARTE 2

Hasta aquí, todo debería funcionar y ser una aplicación independiente.

Ahora vamos a trabajar el concepto de Interfaces.

1. En el paquete **maestre.general.interfaces**

    a. Para ello, **crea un interfaz** llamado **Reparable** que contenga un método:

    public void reparar ();

Ahora, haz que las clases **Portatil** y **Sobremesa**, implementen dicho interfaz. (para ello tienes que realizar una pequeña modificación)

Si lo has hecho bien, te obligará a implementarles a esas clases el método **reparar** () que lo único que hará es mostrar unos asteriscos y un mensaje indicando que el ordenador se está reparando.

2. En el paquete **maestre.electrodomesticos**
    a. Crea una clase llamada **Aspirador**, con algún atributo y método que te inventes, pero que **implemente la interfaz Reparable**. Eso te obligará a implementarle el método **reparar**(). En este caso, puede hacer otra cosa diferente al reparar de los ordenadores, en este caso, muestra un mensaje, pero diferente, en vez de asteriscos elige otro formato diciendo que el aspirador se está reparando.
3. En el paquete **maestre.general.app** Crea una aplicación, llamada **Parte2App** (así, no tenemos que modificar la realizada en la parte 1), además del método main, añade otro método que se así:
    a. Método: **static void reparaTodo(Reparable cosas[])**
    Verás que se puede hacer un array de Reparable, es decir de objetos que implementan esa interfaz. Es una maravilla esto ;-)

    b. Desde el **main**, crea un array de 3 objetos Reparables, en el que precargues 2 Portátiles, 2 Sobremesas y 2 Aspiradores (como ves, estamos mezclando).
    Llama al método reparaTodo, pasándole ese array. (se verá por pantalla que todo se está reparando y eso que son objetos de muy diferente tipo)
4. **Implementa el interfaz Comparable en los portátiles** (pequeño cambio):
    Te obligará a implementar el método compareTo. Un portátil es mayor que otro si las pulgadas son mayores, es el criterio que usaremos. (todo esto está explicado en el anexo)
5. **Crea un array** con 3 portátiles, cuyas pulgadas sean diferentes y esté desordenado. (**muéstralo** desordenado)
6. **Ordena** el array de manera ascendente utilizando el método sort de Java. (al haber obligado a los portátiles, a implementar el compareTo, ya te permite usar el método sort)
7. **Vuelve a mostrar el array**, que ya tiene que salir ordenado.

Criterios de puntuación. Total 10 puntos.

Requisito para evaluar, que el documento demuestre la identificación del alumno, mostrando de fondo en los pantallazos, el aula virtual del curso junto con la identidad. En este caso no hay vídeo como en otros temas, sino un pdf.

Además, en ese documento se deberán realizar **explicaciones de lo construido**, y una prueba de la **ejecución** de **todas las opciones**, que se explique cada apartado en orden que se pueda comprender por parte del lector.

Realizar una buena explicación y prueba del propio programa de todos los apartados por parte del propio alumno, es necesario, y si algún apartado no se ha realizado, se indicará.

En la puntuación, se parte sobre 10 puntos.

PARTE 1: **6 PUNTOS** (0.5 puntos de penalización cada apartado o punto importante que falte, nunca se puede puntuar negativo en esta parte)

PARTE 2: **4 PUNTOS** (0.5 puntos de penalización cada apartado o punto importante que falte, nunca se puede puntuar negativo en esta parte)

La **no realización del documento identificativo**, hará que la práctica no pueda corregirse y su puntuación **será de 0**.

Si el documento, demuestra identidad, pero es **insuficiente** respecto a lo pedido, penalizará en **2 puntos** sobre la nota obtenida.

Recursos necesarios para realizar la Tarea.

- Ordenador personal.
- Sistema operativo Windows o Linux.
- JDK y JRE de Java (preferiblemente últimas versiones)
- NetBeans (últimas versiones preferentemente)

Consejos y recomendaciones.

Dentro de la carpeta de NetBeans, se habrá creado automáticamente la carpeta del proyecto. Comprueba que efectivamente está la carpeta y dentro todo el contenido que has creado.

Para entregar, **comprime en un único** archivo la **carpeta del proyecto** con todo su contenido, junto con el **documento PDF** y **pantallazos con las explicaciones que demuestran la identidad**.

Indicaciones de entrega.

El alumnado debe realizar un **proyecto de NetBeans** llamado **Tarea8_Nombre_Apellidos**, que tendrá que comprimir y entregar junto con el **documento explicativo e identificativo** con las explicaciones y ejecuciones que se realicen y las consideraciones oportunas (dificultades, interpretaciones o aclaraciones que surjan en algún, apartado).

La entrega por lo tanto será un archivo comprimido, llamado **Tarea7_Nombre_Apellidos**, de tal manera que, al descomprimir, se vea descomprima el proyecto de Netbeans y el PDF.