UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Tecnología e Innovación para Labrar el Desarrollo Educativo

Convocatoria ordinaria de Programación I

Examen 2 (20/01/2020) DURACIÓN **3 horas**

Nombre: .....

Apellidos: .....

Hora entrega: ..... Código ordenador: .....

Notas:

- Rellene sus datos personales y el código del ordenador donde realiza el examen, y al finalizar el examen la hora de entrega. Entréguelo al profesor antes de salir.
- Para la realización del examen SÓLO SE PUEDE ACCEDER a la documentación disponible en la página de la asignatura y a la documentación en línea de oracle.com que incluye la especificación de las clases y los tutoriales.
- No se permite llevar consigo ningún dispositivo electrónico durante el examen. Las pertenencias deben guardarse en el lugar indicado del aula y los móviles deben estar apagados.

Enunciado

Se desea construir un conjunto de clases que gestione un concesionario de venta de vehículos. Los objetos a representar son vehículos y extras a añadir a un vehículo. También se desea disponer de la lista de ventas. Se ha decidido crear la interface **`Vendible`** (ya desarrollada) y las clases: **`Vehiculo`**, **`Extra`** y **`Ventas`**. Los vehículos y extras son vendibles. Las ventas representan una lista de objetos vendibles.

Las operaciones de la interface **`Vendible`** son:

Método	Descripción
String dameId()	Devuelve una String como identificador único del elemento vendible.
String dameNombre()	Devuelve una String como nombre del elemento.
int damePVP()	Devuelve un entero con el precio de venta al público.

La clase **`Extra`** representa objetos vendibles y dispone, además, de los siguientes métodos:

Método	Descripción
Extra(String, String, **int**)	Constructor que inicializa un Extra al que se le pasa: una String con el identificador, una String con el nombre y un **int** con el PVP.
String toString()	Devuelve una String con la concatenación del identificador, el nombre y el PVP, separados por un espacio.

La clase **`Vehiculo`** se define como vendible y puede contener un conjunto de extras. Tiene, además, los siguientes métodos:

Método	Descripción
Vehiculo(String, String, **int**)	Constructor que inicializa un Vehiculo al que se le pasa: una String con el identificador, una String con el modelo de vehículo como nombre y un **int** con el PVP. Falta añadir los extras al vehículo.
int damePVP()	Devuelve un entero con el precio inicial más la suma de los extras.
boolean añadeExtra(Extra)	Añade un extra, si no está. Devuelve verdadero o falso según lo haya podido añadir o no.
boolean quitaExtra(Extra)	Se le pasa un extra y lo quita del vehículo. Devuelve verdadero o falso según lo haya podido quitar o no.
boolean estáContenido(Extra)	Devuelve verdadero o falso según el extra esté contenido o no.
Set<Extra> dameExtras()	Devuelve un conjunto con los extras que tiene el vehículo ordenados por PVP, de mayor a menor precio.

String toString()	Devuelve una String con la concatenación del identificador, el nombre y el PVP inicial (sin los extras), separados por un espacio y, si hay extras, sigue un espacio y, entre paréntesis, los extras separados por comas seguidas de un espacio (`", "`), (ver ejemplo en main).

Los objetos de la clase **`Ventas`** mantienen una lista con las ventas realizadas, además de obtener información sobre éstas. Los métodos que tiene son:

Método	Descripción
void añadeVenta(Vendible)	Añade un vendible como nueva venta realizada.
int totalVentas()	Devuelve un entero con el total del PVP de ventas realizadas.
List<Vendible> listaVentas()	Devuelve una lista con los objetos vendidos en el orden añadidos.
String másVendido()	Devuelve el identificador del Vendible más vendido. En caso de existir varios con la máxima frecuencia se devolverá el de menor identificador. En caso de no existir ninguna venta se devolverá `null`.
String toString()	Devuelve una String con un listado numerado (un Vendible por línea) de las ventas realizadas, con la concatenación de todas las ventas realizadas separadas por saltos de línea (`"\n"`) precedidos por el número de venta seguido de un cierra paréntesis y un espacio en blanco, `") "`, (ver ejemplo en main).

Ejercicio 1 (50%)

Realice el desarrollo de las distintas clases según los requisitos descritos en el enunciado añadiendo los métodos de la clase **`Object`** y comparadores que cree oportunos para su correcto funcionamiento en el resto de la aplicación, usando los contenedores de la librería de Java más adecuados. Las clases con el código a desarrollar deben pertenecer al paquete: `"concesionario.source"`.

Ejercicio 2 (50%)

El ejercicio consiste en desarrollar una aplicación con interfaz gráfica que use las clases **`Extra`**, **`Vehiculo`** y **`Ventas`** anteriores. La aplicación tendrá:

- Una ventana principal donde se muestra el listado de vehículos vendidos y un botón para añadir vehículos a las ventas.
- Al pulsar el botón de "Añadir vehículo" debe aparecer un diálogo para introducir los datos del nuevo vehículo incluyendo sus extras.
- Para detallar los extras de un vehículo en el diálogo anterior se podrá pulsar un botón "Añadir extra al vehículo" que abra otro diálogo para añadir un nuevo extra al vehículo.

Las clases **`Extra`**, **`Vehiculo`** y **`Ventas`** se suministran desarrolladas. La ventana principal, que se suministra parcialmente realizada, se desarrollará en la clase **`ConcesionarioGUI`**. Las clases con el código a desarrollar deben pertenecer al paquete: `"concesionario.gui"`.