Innovación en Formación Profesional

Módulo Profesional 03:

Programación II

Actividad Evaluable UF4

CICLO FORMATIVO DE GRADO SUPERIOR EN

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

MODALIDAD ONLINE

Fábrica OO

Objetivos

- Seleccionar y emplear lenguajes, herramientas y librerías, interpretando las especificaciones para desarrollar aplicaciones multiplataforma con acceso a bases de datos
- Identificar los cambios tecnológicos, organizativos, económicos y laborales en su actividad, analizando sus implicaciones en el ámbito de trabajo, para mantener el espíritu de innovación

Competencias asociadas:

- Configurar y explotar sistemas informáticos, adaptando la configuración lógica del sistema según las necesidades de uso y los criterios establecidos

Metodología

- Preparación individual
- Para la realización del ejercicio se deberán visualizar todos los videotutoriales del curso.

Entrega

27 de octubre del 2020.

Se deberá entregar un único proyecto en Java o C++ comprimido en **Zip** con la nomenglatura

MP03_PII_UF04_FabricaPOO.zip

(el resto de requisitos de entrega se describen en la sección descripción de la actividad).

Dedicación estimada

10 horas

Documentos de referencia

Videotutoriales de la UF.

Libro de referencia de la asignatura

Resultados de aprendizaje

- Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

Criterios de evaluación

- Se han identificado los bloques que componen la estructura de un programa informático.
- Se han utilizado entornos integrados de desarrollo.
- Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
- Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
- Se han introducido comentarios en el código.

Descripción de la actividad

Aprenderás a trabajar con programación orientada a objetos, sabrás realizar diferentes clases y organizarlas en paquetes para posteriormente utilizarlas como librerías.

- Se deberá entregar con la nomenglatura y formato correcto.
- Todas las variables deberán comenzar por minúscula .(-0.5 puntos por cada variable con un nombre erróneo)
- Todas las funciones deberán comenzar por minúscula (-0.5 puntos por cada nombre de función erróneo).
- Se interpretará que el usuario escribirá un dato del tipo de la variable donde se almacenará, es decir, si se pide un entero el usuario siempre escribirá un entero nunca un string…
- El ejercicio deberá poder ejecutarse sin problema, es decir, no deben existir errores en la compilación (en caso de tener errores el código el ejercicio evaluable será suspenso).
- Nota: Es mejor entregar pocos apartados que estén bien, que muchos y no compile.

Desarrollo de la actividad

La empresa **MATERIALCONTS** desea crear un programa que gestione:

Los **productos** de la empresa

Los **vehículos** de la empresa

Los **edificios** de la empresa.

A día de hoy no poseen una base de datos donde almacenar toda la información, por lo que nos piden que creemos la estructura de clases para poder gestionar toda la información que desean.

Desean que la organicemos en paquetes para que posteriormente las puedan utilizar en futuros proyectos que hagan.

Nota importante: Todas las clases que se definan en el ejercicio tendrán que implementar el método print para mostrar por la consola toda la información de los atributos del objeto.

Vehículos **(2 puntos)**

La empresa desea gestionar la información de todos sus **vehículos**, posee vehículos de tres tipos:

- Gasolina
- Gasoil
- Eléctricos.

Cada uno de ellos tienen características comunes y cada uno de ellos posee algunas características propias.

Todos los vehículos tienen una marca, velocidad actual, está arrancado(booleano), un precio, un peso, un color y una longitud.

Por otro lado, los vehículos de gasolina poseen las características contaminación y tamaño depósito exclusivas

También los vehículos eléctricos poseen las características potencia y velocidad máxima que no poseen los otros dos tipos.

Todos los vehículos sea cual sea su tipo **DEBEN OBLIGATORIAMENTE implementar** de una interfaz llamada **GestionVehiculo** los métodos arrancar(), frenar(), acelerar(), parar() ya que todos los vehículos deben realizar estas acciones independientemente del tipo.

- La implementación del método **acelerar** debe devolver un booleano indicando si la operación se ha realizado con éxito (true si el coche esta arrancado y false en caso contrario) y se posee un único parámetro de tipo float que es el incremento de la velocidad actual
  boolean acelerar(float)
- La implementación del método **frenar** debe devolver un booleano (true si el coche esta arrancado y false en caso contrario) y se posee un único parámetro de tipo float que es el decremento de la velocidad actual, en caso de que la velocidad sea inferior a 0, cambiar la velocidad actual a 0.
  boolean arrancar(float)

- La implementación del método **arrancar** debe devolver un booleano (true si el coche no esta arrancado y false en caso contrario) y no posee parámetros
  boolean arrancar()

- La implementación del método **parar** debe devolver un booleano (true si el coche está arrancado y false en caso contrario) y no posee parámetros.
  boolean parar()

Edificios **(2 puntos)**

Por otro lado, se desean gestionar los edificios que componen la empresa. Hay 3 tipos de edificios:

- Fabrica
- Almacén

- Oficina.

Todos los tipos de edificios tienen unas características comunes, anchura, altura, profundidad, material, tipo de edificio en relación a su funcionalidad, precio de mercado y color.

Los edificios poseen una serie de funcionalidades una de ellas es **costePintura** a la cual se le pasa como parámetro el color y el precio de pintura por metro y te devolverá un float con el coste que supondría pintar el edificio (coste= precio por metro * anchura * altura* profundidad).

En caso de que el precio de coste sea inferior o igual a 0 la función devolverá un -1

La funcionalidad **costePintura** estará sobrecargada, existirá otro método con el mismo nombre y devolución información pero se le pasará un parámetro más de tipo String, con los posibles valores ladoanchura o ladoprofundidad.. Si el valor es ladoanchura el coste sería coste= precio por metro * anchura * altura en caso de ser ladoprofundidad el coste sería coste= precio por metro * altura* profundidad, en caso de introducir otro valor la función devolverá -1.

En caso de que el precio de coste sea inferior o igual a 0 la función devolverá un -1

Por otro lado se desea crear en la clase padre Edificio un método abstracto llamado funcionalidadEdificio a la cual no se le pasa ningún parámetro y devuelve un String con la información del tipo de edificio en concreto.

La funcionalidad de cada edificio es la siguiente:

En la fábrica se crea el producto.

En la oficina se etiqueta el producto

En el almacén se guarda el producto para posteriormente venderlo.

Productos **(2 puntos)**

También desean gestionar los productos de la empresa. La empresa posee 3 tipos de productos:

- Ventanas
- Puertas
- Sillas.

Todas las empresas poseen las siguientes características: identificador único (ningún producto tiene el mismo identificador), nombre, tipo, anchura, profundidad, altura, color y Edificio (es el edificio donde se encuentran).

Empresa **(2 puntos)**

La empresa posee edificios, coches y productos

Poseen 4 coches: 2 eléctricos, uno de gasolina y uno diesel.

La empresa posee 3 edificios, una fábrica, un almacén y una oficina.

Poseen 6 productos como máximo simultáneamente: 2 puertas, 2 sillas y 2 ventana todos ubicados en la fábrica

En la empresa se realizan varias funcionalidades:

boolean crearProducto(Producto): se añade un nuevo producto a la empresa, que se le pasa como parámetro, si tiene hueco y el identificador no existe, en caso de añadirlo devolverá true, en caso contrario devolverá false (tanto si no tiene hueco, como si el identificador existe).

boolean venderProducto(Producto) : se vende un producto de la empresa que se le pasa como parámetro, si existe (existe su identificador), lo borra y devuelve true, en caso que no exista devuelve false.

Nota: Un producto se podrá vender solo si se encuentra en el almacen.

boolean cambiarUbicaciónProducto(Producto, Edificio) se pasa como parámetro el producto que se desea cambiar la ubicación y el edificio al cual se desea cambiar, si el producto existe (coinciden los identificadores) se modificará el edificio del producto y devolverá un true, en caso contrario devolverá false

Importante: La case Fabrica, Almacén y Oficina son Edificios ya que serán hijas suyas, por lo tanto se les puede pasar como parámetro a este método.

Creación de la empresa OO **(2 puntos)**

Vehículos:

Poseen 4 coches: 2 eléctricos, uno de gasolina y uno diesel.

Los vehículos eléctricos deben están arrancados e ir a una velocidad de 50 km/h.

Los otros dos vehículos deben frenar tras ir a una velocidad de 80km/h.

Edificios:

La empresa posee 3 edificios, una fábrica, un almacén y una oficina.

La funcionalidad de cada edificio es la siguiente:

En la fábrica se crea el producto.

En la oficina se etiqueta el producto

En el almacen se guarda el producto para posteriormente venderlo.

Calcular el precio de pintar la fábrica completa de color verde y precio por metro 30 euros.

Productos:

Poseen 6 productos como máximo simultáneamente: 2 puertas, 2 sillas y 2 ventana todos ubicados en la fábrica

Poseen una puerta con identificador 1 y otra con el identificador 2

Poseen una silla con el identificador 3 y otra con el 4

Poseen una ventana con el identificador 5 y otra con el 6

Cambiar la ubicación de las sillas y llevarlas a la oficina.

Cambiar la ubicación de las puertas y llevarlas al almacén.

Vender la puerta con el identificador 1

Vender una silla con el identificador 3

Crear una puerta nueva con el identificador 2

Crear una puerta nueva con el identificador 10