Parcial febrero 2021

Para evaluar este examen se tendrá en cuenta tanto el correcto funcionamiento de los ejercicios, como la eficiencia de los mismos, por esa razón un ejercicio podrá obtener la máxima nota siempre y cuando tenga en cuenta ambas premisas.

Los ejercicios deben estar resueltos en una página HTML para demostrar su correcto funcionamiento, pudiéndose utilizar la consola, además se debe utilizar una codificación estricta de JavaScript. **Si un ejercicio no está demostrado, el ejercicio se puntuará con un máximo de la mitad de la nota de dicho ejercicio.**

En cada ejercicio se deberá utilizar los métodos de los diferentes objetos del API de JavaScript para su resolución. En el caso de implementar funcionalidad que ya exista en el API, el ejercicio se verá penalizado por dicha circunstancia.

Por último, no se pueden presentar errores en las pruebas de los ejercicios. Errores de sintaxis o no captura de excepciones implicarán la reducción de la nota o la perdida total de la puntuación.

Instrucciones de entrega

Crea una carpeta con tu nombre, ejemplo “LizanoMontalvoPablo”, y dentro de ella crea una carpeta por cada ejercicio, ejemplo “ejercicio1”, “ejercicio2”, etc, las cuales deberán contener la resolución de cada ejercicio. Empaqueta la carpeta principal en un ZIP y súbelo a Delphos.

Si lo ves necesario, explica el ejercicio en un párrafo de la página HTML que hayas creado.

1. Ejercicio único

Partimos del modelo de datos creado para el ejemplo **ShoppingCart** proporcionado como contenido de la UT04. Este ejemplo disponía de una estructura de clase abstracta para **Producto** de la cuál heredaban las clases **Camera**, **Tablet**, **Laptop** y **Smartphone**.

En este ejercicio debemos crear una estructura similar a la de la **práctica 4** con una clase **Manager** que nos permita relacionar una serie de objetos de los tipos previamente citados, organizados en categorías.

La funcionalidad a implementar se describe en los siguientes puntos, cada una de las cuales debe estar correctamente demostrada en una función de testeo.

1.1. Clase Category

Implementa una clase **Category** con dos propiedades **title** y **url** obligatorias e introducidas como parámetros en el constructor, y una propiedad **description** optativa, garantizando que no sean vacíos.

1.2. Propiedad url

Todos los objetos de las subclases de **Product** debe incluir un atributo **url**.

1.3. Objeto Manager

Crea una estructura para instanciar un objeto Manager utilizando un patrón **Singleton**.

1.4. Método addCategory()

Crea un método que permita añadir un objeto **Category** al Manager. Si existe ya una categoría con el atributo **title** debe lanzarse una excepción.

1.5. Método addProduct()

Crea un método que añada un objeto **Product** asociado a un objeto **Category**. Si el objeto **Category** no existe en el Manager se debe añadir también. No puede existir en la misma categoría un producto con el mismo número de serie.

1.6. Método removeProduct()

Elimina un objeto **Product** dentro de un objeto **Category**. Si el objeto no existes en la categoría se debe lanzar una excepción.

1.7. Estructura de excepciones

Crear una estructura jerárquica de excepciones donde se pueda personalizar el mensaje de cada una de ellas con sus respectivos parámetros.

- ManagerException. Excepción genérica del Manager.
    - CategoryExistsException. Indica que el objeto **Category** ya existe en el manager. Muestra en el mensaje el título de la categoría.
    - CategoryNotExistException. Indica que el objeto **Category** no existe en el manger. Muestra en el mensaje el título de la categoría.
    - ProductInManagerNotExistException. Indica que el producto no existe asociado a la categoría. Debe mostrar en el mensaje el número de serie y el título de la categoría.

1.8. Iterador de categorías

Crea una propiedad que devuelve un iterador que muestre los objetos Category almacenados en Manager ordenados por el título.

1.9. Iterador de productos

Crea un método que devuelva un iterador con los productos de una determinada categoría ordenados por el precio de mayor a menor.

1.10. Método toString()

Genera un método toString() que muestre el contenido del Manager utilizando los iteradores de categorías y de productos.