Certified Tech Developer
The Ultimate Degree

Programación Imperativa

Examen final de Programación Imperativa

Objetivo

Evaluar las habilidades adquiridas durante la cursada en el contexto de la Programación Imperativa utilizando Javascript corriendo en un entorno de Node.

Metodología de evaluación

En base a las consignas requeridas en el examen, se evaluarán los siguientes conceptos sobre el código entregado:

- **FORMA**
    - Que el código esté prolijo e implemente buenas prácticas
    - Que las variables, métodos y funciones tengan nombres descriptivos
    - Que se utilicen nombres en español o en inglés pero no ambos
    - Que se utilice camelCase donde corresponda

- **LÓGICA**
  - Que la lógica corresponda con lo que solicitan las consignas
  - Que se utilicen los métodos más adecuados para resolver cada problema
- **FUNCIONAMIENTO**
  - Que el código funcione correctamente, sin arrojar errores
  - Que el código produzca el resultado esperado a partir de los datos suministrados

Duración y entrega

El examen tendrá una duración de 2 horas. Durante los primeros 15 minutos se explicará el funcionamiento del mismo y durante los siguientes 105 minutos tendrán tiempo para resolverlo. Las entregas realizadas luego del tiempo estipulado no serán tenidas en cuenta.

Al terminar el examen deberán entregarlo haciendo uso de **este formulario** C3.

¡Les deseamos lo mejor en el examen! 💪

Consignas

En este examen estaremos modelando una carrera de autos, teniendo en cuenta la inscripción de los mismos en base a determinadas condiciones y generando la lógica necesaria para simular el resultado de la carrera, determinando en qué puestos quedarán los corredores.

Los datos de los autos los obtendremos de un archivo con formato JSON que contendrá un array de objetos literales que representarán los autos.

Consignas

1. Obtener el listado de posibles participantes

Tomando como base el siguiente archivo JSON, deberán leer el archivo y parsearlo para obtener el listado de autos que desean participar de la carrera.

Resultado esperado: variable conteniendo un array con todos los autos disponibles.

2. Crear un objeto literal que represente la carrera

Este objeto contendrá, una propiedad con el listado de autos y las **funcionalidades** que nos solicitan a continuación:

A. Agregar una propiedad llamada ***autos*** que contenga los autos obtenidos en el punto anterior.
B. Agregar una propiedad llamada **autosPorTanda** que contenga el valor **4**. Este valor representará la cantidad máxima de autos por tanda.
C. Agregar un método **autosHabilitados**, que permita consultar los autos habilitados, es decir, que devuelva una lista de los autos que no estén sancionados.

   Resultado esperado: un array con los autos habilitados para correr.

D. Agregar un método **listarAutos** que reciba como parámetro un array de autos y los imprima por consola.

   Este método deberá imprimir por cada elemento:

   - La placa o patente.
   - El piloto
   - El peso del auto
   - *"sancionado"* → En caso de ser true la propiedad sancionado
     "puede correr" → Caso contrario

   Resultado esperado: un mensaje por consola por cada auto con el siguiente formato: *"Patente: XXXXXX,Piloto: XXXXXX, peso en kg: XXX, estado: sancionado / puede correr"*.
   Ej 1: "Patente: ABC123, Piloto: Monah Lyal, peso en Kg: 267, estado: puede

correr”

Ej 2: “Patente: EFG567,Piloto: Juan Corredor, peso en Kg: 357, estado: sancionado”

E. Agregar un método **buscarPorPatente** que permita buscar el auto en función de su patente.
    - Este método debe devolver un auto en caso de encontrar la patente en el array de la propiedad autos.

Resultado esperado: el auto que coincida con la patente ingresada como parámetro.

F. Agregar un método **buscarPorAnguloDeGiro** que permite filtrar **los autos habilitados**, siempre y cuando su propiedad angulo de giro sea menor o igual al ángulo de giro enviado como argumento.
    - Este método debe usar **autosHabilitados.**

Resultado esperado: un array con los autos que cumplan las condiciones mencionadas.

G. Agregar un método que permita **ordenar por velocidad** de menor a mayor según la velocidad de cada auto y devolver un array ordenado.

Resultado esperado: un array de autos ordenado por velocidad de menor a mayor.

H. Agregar un método **generarTanda** que permita retornar un array de autos, que cumpla con las siguientes condiciones:
    - únicamente autos habilitados

- Angulo de giro del auto menor o igual a un valor enviado como argumento.
- peso del auto menor o igual a un valor enviado como argumento

Resultado esperado: un array con los autos que cumplan las condiciones especificadas y tenga como máximo la cantidad de autos expresada en la propiedad **autosPorTanda** del objeto carrera.

I. Agregar un método que permita **calcularPodio**, el mismo deberá calcular al ganador y los siguientes dos puestos en función de su puntaje.
   - El podio debe surgir a partir de la **tanda generada**.
   - Los primeros tres autos con mayor puntaje conformarán el podio

   Resultado esperado: imprimir por consola el piloto y el puntaje del podio diferenciando primer, segundo y tercer puesto.

   Ej:

   "El ganador es: Leandro Ezequiel, con un puntaje de: 70;
   el segundo puesto es para: Martin Cejas, con un puntaje de: 55
   y el tercer puesto es para: Nicolas Lopez, con un puntaje de: 52"