Certified Tech Developer
The Ultimate Degree

Programación Imperativa

Recuperación de 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
- Que utilice la forma adecuada para mostrar por consola e invocación de cada método.

- **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

La recuperación del 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**.

¡Les deseamos lo mejor en el examen! 💪

Contexto Consigan

En este examen estaremos modelando funcionalidades para un sistema de tickets de mesa de ayuda del área de soporte IT de una empresa, teniendo en cuenta las características de los mismos en base a determinadas condiciones y generando la lógica necesaria para simular el proceso de una mesa de ayuda.

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

Consignas

1. Obtener el listado de tickets

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

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

2. Crear un objeto literal que represente la mesa de ayuda

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

A. Agregar un método **buscarTicket**, que permite consultar un ticket enviando como parámetro el nroTicket deseado.

   Resultado esperado: el ticket según el parámetro enviado.

B. Agregar un método **listarTickets** que reciba como parámetro un array de tickets y los imprima por consola.

   Este método deberá imprimir por cada elemento:

   - número de Ticket
   - usuario
   - prioridad
   - *"resuelto"* → En caso de ser true la propiedad estaResuelto
     "pendiente" → Caso contrario
   - minutos de espera

Resultado esperado: un mensaje por consola por cada ticket con el siguiente formato: *"Nro Ticket: XXXXXX, usuario: XXXXX,prioridad: XXX, estado: resuelto / pendiente , Minutos: XXX"*.
Ejemplo:

nroTicket: 10, usuario: Regina Medina, prioridad: media, estado: Resuelto, min de tiempo de espera: 175.

nroTicket: 12, usuario: Gina Velez, prioridad: baja, estado: Resuelto, min de tiempo de espera: 190.
ATENCIÓN: *este método debe ser usado para listar todos los siguientes métodos que nos retornen un array de tickets.*

C. Agregar un método **ticketsPendientes** que permita filtrar tickets sin resolver, esto es indicado por la propiedad estaResuelto.

Resultado esperado: Este método debe devolver un array de tickets pendientes.

D. Agregar un método **buscarPorPrioridad** que permita filtrar **los tickets pendientes**, siempre y cuando la prioridad sea igual al parámetro recibido.
   - Este método debe usar **ticketsPendientes .**
   - Este método debe recibir como parámetro un string que representa la prioridad del ticket. *("alta"), ("media") o ("baja")*

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

E. Agregar un método que permita **ordenar por tiempo de espera** de mayor a menor según la propiedad **minutosDeEspera** de cada ticket que se encuentre pendiente y devolver un array ordenado.

   - Este método debe usar **ticketsPendientes.**

   Resultado esperado: un array de tickets ordenado por tiempo de espera de mayor a menor.

F. Agregar un método **informarAlUsuario** que reciba por parámetro un nro de ticket el cual se informará resuelto, es decir que imprima por pantalla un mensaje del ticket resuelto.

   Resultado esperado: un mensaje por consola con el siguiente formato: *"Sr/a: XXXXXX, Se informa que su ticket Nro: XXX, prioridad: XXX, está resuelto.*