15/03/2025

EXAMEN RECUPERACIÓN (UD1-UD4)

Los RA's que evaluaremos son:

RA1 – RA2 – RA3 – RA4 - RA5

Aventura de Super Mario

El objetivo es trabajar la **Programación Orientada a Objetos (POO) en JavaScript**.

Parte 1: Implementación de Clases (3,5 puntos)

Se implementarán las siguientes clases con validaciones.

1. **Clase Personaje** (0,75 puntos)
    - **Atributos privados:**
        - nombre (string): Debe ser una cadena no vacía.
    - **Métodos:**
        - constructor(nombre): Inicializa el atributo validando que el nombre no esté vacío.
        - toString(): Devuelve una representación textual del personaje.
        - Getters y setters con validaciones que lancen errores si los valores no cumplen las restricciones.
2. **Clase Heroe (hereda de Personaje)** (0,75 puntos)
    - **Atributos adicionales privados:**
        - habilidadEspecial (string): Habilidad única del héroe, según la siguiente tabla:
            - Mario → Lanzar Fuego
            - Luigi → Salto Alto
            - Peach → Flotar
            - Toad → Velocidad Extra
    - **Métodos adicionales:**
        - Getters y setters con validaciones.
        - toString(): Devuelve una representación textual del héroe.
3. **Clase Enemigo (hereda de Personaje)** (0,75 puntos)
    - **Atributos adicionales privados:**
        - debilidad (string): Debilidad del enemigo, según la siguiente tabla:
            - Bowser → Fuego
            - Goomba → Salto
    - **Métodos adicionales:**
        - Getters y setters con validaciones.
        - toString(): Devuelve una representación textual del enemigo.
4. **Clase Aventura** (1,25 puntos)
    - **Atributos privados:**
        - nombreMundo (string): Debe ser una cadena no vacía y de al menos 5 caracteres.
        - heroes (array de objetos Heroe): Debe contener al menos **2 héroes**.
        - enemigo (objeto Enemigo): Solo puede haber **un enemigo** por aventura.
    - **Métodos:**
        - constructor(nombreMundo, heroes, enemigo): Inicializa los atributos validando las reglas.

- Getters y setters con validaciones.
- toString(): Devuelve una descripción de la aventura, incluyendo los héroes y el enemigo.

Parte 2: Registro y Validaciones (2,4 puntos)

Se proporcionará un archivo index.html, styles.css e imágenes. Los alumnos solo deben centrarse en la lógica con JavaScript.

El formulario ya incluye:

- Campo "Nombre del Mundo" (mínimo 5 caracteres, validado en JavaScript).
- Selección de héroes (mínimo 2 seleccionados, usando checkboxes).
- Selección de enemigo (solo 1 seleccionado, usando radio buttons).
- Botón "Iniciar Aventura".

Las **validaciones deben realizarse en JavaScript**, mostrando los errores **dentro del formulario** (sin alert()).

Reglas de validación:

- El nombre del mundo debe tener al menos 5 caracteres.
- Deben seleccionarse al menos 2 héroes.
- Debe seleccionarse un enemigo.
- Si hay errores, el mensaje debe mostrarse en rojo encima del botón "Iniciar Aventura".
- Si los datos son correctos, se mostrará un mensaje de éxito en verde.

Parte 3: Interactividad y Validaciones (4,1 puntos)

Al hacer clic en "Iniciar Aventura":

- Se crearán instancias de Heroe y Enemigo basadas en la selección del usuario.
- Se creará una instancia de Aventura y se mostrarán los detalles en la consola con console.log().
- Si los datos son válidos: Mostrar un mensaje de éxito en verde en el formulario indicando que la aventura ha comenzado.

Entrega

Los alumnos deben entregar los siguientes archivos (HTML y CSS ya están dados):
personaje.js → Implementación de la clase Personaje.
heroe.js → Implementación de la clase Heroe.
enemigo.js → Implementación de la clase Enemigo.
aventura.js → Implementación de la clase Aventura.
index.js → Lógica de validación e interacción con el formulario.