Recuperatorio Programación Básica 1 - Q1 - C64 - TN - 2023

Plataforma de juegos

Nos han contratado para desarrollar un producto software que permita gestionar juegos obtenidos desde una plataforma de juegos.
En esta primera entrega, realizaremos un desarrollo que nos permite ver los juegos disponibles en la plataforma, adquirir uno o varios juegos, y poder jugar.
Para lograr esto, deberemos completar las siguientes clases del proyecto suministrado:

Clase de prueba PlataformaJuegos

Será la clase que contenga al método main y deberemos desarrollar lo siguiente para garantizar el correcto funcionamiento:

- **private static void registrarme(Plataforma plataforma):** Pide el ingreso de los datos necesarios para crear un usuario. Debe verificar que el correo sea válido y que no exista otro usuario registrado en la plataforma con dicho correo (se sugiere buscar el usuario por correo). Una vez obtenidos los datos, se procede a registrar el usuario en la plataforma mostrando un mensaje en caso de éxito y otro en caso de error.
- **private static void iniciarSesionEn(Plataforma plataforma):** Pide el ingreso de credenciales (usuario y contraseña). Verifica si con esas credenciales se puede iniciar sesión en la plataforma. Si el inicio de sesión es exitoso, se muestra un mensaje de éxito y seguido el menú del usuario (método menuUsuario()), caso contrario, se muestra el mensaje de error: "Usuario y/o contrasenia invalido".
- **private static void menuUsuario(Usuario usuario, Plataforma plataforma):**
    - **case MIS_JUEGOS:** Obtiene los juegos del usuario y los muestra. Solicita el ingreso de un número entero mostrando un mensaje. Dicho número corresponde con el ID de algún juego (no validamos que sea correcto en esta primera versión). Se debe jugar al juego ingresado (ver método para jugar). Si se ingresa 99 se sale del menú actual.
    - **case TIENDA:** Por defecto contaremos con 10 juegos pre cargados para poder operar. Se solicita el ingreso de un número que deberá corresponderse con alguno de los IDs de los juegos. Si el número ingresado se encuentra entre 1 y 10 inclusive, entonces se procede a adquirir el juego. Se sugiere obtener desde la plataforma el juego por su ID y luego agregarlo a los juegos del usuario. Es necesario mostrar un

Nombre y Apellido DNI FIrma

mensaje de éxito en caso de adquirir el juego. Si no se pudo adquirir el juego, entonces mostrar el mensaje: "No fue posible agregar el juego, verifica que no hayas adquirido antes."

- **case JUEGO_MAS_JUGADO_POR_CATEGORIA:** Mostrar un título que indique para qué categoría se mostrará el juego mas jugado y luego mostrar el juego mas jugado de esa categoría.

Clase Juego

Esta clase representará los juegos que podremos adquirir y jugar desde la tienda. Cada juego tiene un id entero numérico, un nombre, una cantidad de horas jugadas (número con decimales) y una categoría.
Se deberá completar la clase para que el software funcione correctamente. Tener presente que se nos solicita mostrar juegos, con lo cual necesitaremos algún método que nos permita visualizar la información correspondiente.

Clase Usuario

Dicha clase nos representará a un usuario que puede adquirir juegos desde la tienda. Un usuario puede adquirir muchos juegos, siempre y cuando, no lo haya adquirido antes (no se admiten juegos duplicados). De cada usuario necesitamos saber su nombre, apellido, correo, contraseña y tener algo que nos permita guardar los juegos adquiridos desde la tienda. Permitiremos un máximo de 1000 juegos por usuario en esta primera versión.
Se deberán entonces completar los siguientes métodos:

- **public boolean agregarAMisJuegos(Juego juego):** Suma un juego a los juegos del usuario. Verificar no tener el juego adquirido (Ver el metodo tengoElJuegoDe()). En caso de no tenerlo, se agrega el juego suministrado a los juegos del usuario.
- **public boolean tengoElJuegoDe(int id):** Verifica si tengo un juego con el id suministrado en mis juegos. Devuelve verdadero en caso de poseer el juego.
- **public Juego obtenerJuegoMasJugadoPorCategoria(Categoria categoria):** Revisa los juegos que cumplen con la categoría suministrada y obtiene el juego mas jugado de dicha categoría. Si no existe un juego para esa categoría, devuelve null.
- **public Juego[] obtenerJuegosOrdenadosPorCategoria():** Obtiene los juegos del usuario ordenados por categoría.
- **public void jugarAlJuegoDe(int id):** Revisa entre los juegos del usuario si alguno tiene el id suministrado. Si lo encuentra, le agrega 1 hora y media (1.5) a

la cantidad de horas jugadas. Siempre debería encontrar el juego con el id que llega como parámetro.

Clase Plataforma

Será la clase que contenga la lógica del producto software. Tiene la posibilidad de almacenar usuarios que operan en la plataforma (los que se registran) y juegos que estarán disponibles para ser adquiridos por los usuarios registrados. No se permite que existan 2 usuarios con el mismo correo. Deberemos completar los siguientes métodos:

- **public Usuario buscarUsuarioConCorreo(String correo):** Busca un usuario entre los usuarios de la plataforma que tenga el correo suministrado. Si encuentra un usuario con ese correo, lo devuelve. Si NO encuentra un usuario con ese correo, entonces devuelve null.
- **public Usuario iniciarSesion(String correo, String contrasenia):** Obtiene un usuario buscandolo por su correo. Si existe, verifica que la contraseña sea correcta. Si así es, devuelve el usuario. Si ningún usuario tiene ese correo o la contraseña no coincide, debe devolver null.
- **public boolean esValidoEl(String correo):** Validar el correo. Para que el correo sea válido, tiene que tener un carácter arroba ('@') y terminar en ".com". Devuelve verdadero en caso de que el correo suministrado cumpla con estos puntos.
- **public boolean registrarUsuario(Usuario usuario):** Agrega un usuario a la plataforma (fue validado previamente). En caso de éxito, devuelve verdadero.
- **public Juego obtenerJuegoPorSuId(int id):** Busca entre los juegos de la plataforma si alguno tiene el id suministrado. Si hay un juego con ese id, se devuelve el juego, caso contrario, se devuelve null.

Lineamientos

- No es posible modificar la firma de los métodos existentes, pero es posible agregar todos los métodos que se consideren necesarios.
- Se pueden aplicar las mejoras que considere necesarias, siempre cumpliendo el punto anterior.
- Se deberán completar los métodos y código faltante para que el juego funcione correctamente.
- Deberá ajustar el nombre del proyecto suministrado indicando sus apellidos y nombres en donde dice “ApellidosNombres”: PB12023Q1C64TNRecuperatorio-**ApellidosNombres.**

Nombre y Apellido DNI FIrma

- La versión de java del proyecto deberá ser ajustada a la del laboratorio.

Condiciones de aprobación

Se considerará desaprobado todo proyecto que:
a. no compile, o,
b. no se identifiquen los atributos necesarios para resolver lo solicitado, o,
c. los tipos de datos elegidos para los atributos no sean los adecuados, o,
d. no cumpla con las condiciones mínimas descritas a continuación.

Se considerará aprobado aquel examen que cumpla con el 70% de puntaje.

Clase de prueba PlataformaJuegos	registrarme(Plataforma plataforma)	10%
	iniciarSesionEn(Plataforma plataforma)	5%
	menuUsuario(Usuario usuario, Plataforma plataforma)	10%
Clase Juego	Constructor, getters y setters necesarios, método para mostrar detalles del juego	5%
Clase Usuario	Constructor, getters y setters necesarios	5%
	agregarAMisJuegos(Juego juego)	10%
	tengoElJuegoDe(int id)	5%
	obtenerJuegoMasJugadoPorCategoria(Categoria categoria)	10%
	obtenerJuegosOrdenadosPorCategoria()	5%
	jugarAlJuegoDe(int id)	10%
Clase Plataforma	buscarUsuarioConCorreo(String correo)	5%
	iniciarSesion(String correo, String contrasenia)	5%
	esValidoEl(String correo)	5%
	registrarUsuario(Usuario usuario)	5%

Nombre y Apellido DNI FIrma

	obtenerJuegoPorSuId(int id)	5%
Total		100%

Nombre y Apellido DNI FIrma