

Aplicación: Videojuego.

Tabla de Contenidos

Clase Videojuego.	1
Clase Jugador.	2
Clase VideojuegoEstrategia.	2
Interfaz Plataforma.	3
Clase PC.	3
Clase Consola.	4
Clase Movil.	4
Pruebas unitarias	5
Test pruebaGetPrecio()	5
Test pruebaGetPerfilJugador()	5
Test pruebaSetIdJugador()	5
Test pruebaAumentarNivel()	5
Tests pruebaEsPortatil1() y pruebaEsPortatil2()	5
Diagrama de Clase.	6

Clase Videojuego.

Descripción: Clase con la que podemos jugar juegos, registrar jugadores, etc. Esta clase es la clase origen de todo el programa y todas las funciones de este vienen de la clase Videojuego.

Atributos:

nombre: Contiene el nombre del videojuego.

precio: Contiene el precio de venta del videojuego.

fechaSalida: Contiene la fecha en la que el juego se estrena mundialmente y la gente lo puede comprar.

genero: Contiene el género del juego, este indica su tipo y al hacerlo también podemos deducir de qué maneras se puede jugar.

idJugador: Es un código único que identifica al jugador.

Métodos:

getNombre(): Devuelve el nombre del juego.

getPrecio(): Devuelve el precio en venta del juego.

getFechaSalida(): Devuelve la fecha en la que se puso en venta el juego.

getGenero(): Devuelve el género al que pertenece el juego.

getIdJugador(): Devuelve el código único de cada jugador.

setIdJugador(idJugador:Jugador): Permite asignar el código único de cada jugador al videojuego.

mostrarDetalles(): Muestra por pantalla todos los atributos asignados del juego.

getPerfilJugador(j:Jugador): Muestra por pantalla el perfil del jugador especificado con su ID.

Clase Jugador.

Descripción: Esta clase es la que identifica a cada jugador dentro del videojuego o hace uso de la clase videojuego.

Atributos:

nombre: Contiene el nombre del jugador.

id: Contiene el identificador único del jugador.

nivelCuenta: Contiene el nivel de la cuenta del jugador, este será más o menos alto dependiendo de la cantidad de tiempo que haya jugado el jugador.

Métodos:

getNombre(): Devuelve el nombre del jugador.

getId(): Devuelve el código único del jugador.

getNivelCuenta(): Devuelve el nivel de la cuenta del jugador, este puede aumentar.

muestraPerfil(): Muestra por pantalla el perfil del jugador.

aumentarNivel(): Este método aumentará el nivel de la cuenta del jugador.

Clase VideojuegoEstrategia.

Descripción: Esta clase es un tipo de la clase Videojuego, esta clase por ejemplo es un tipo de juego en el que hay que pensar ciertas estrategias para derrotar al enemigo.

Atributos:

tipoEstrategia: Contiene el tipo de estrategias que se podrán realizar a la hora de jugar.

duracion: Contiene la duración de las decisiones, en este tipo de juegos es importante tener en cuenta la capacidad de decidir rápido y de manera inteligente.

Métodos:

setIdJugador(idJugador:Jugador): Permite asignar el código único de cada jugador al videojuego.

calcularDuración(): Calcula la duración total de la partida y cuanto se tardan en tomar las decisiones.

Interfaz Plataforma.

Descripción: Esta interfaz indica las plataformas en las que podemos jugar nuestro videojuego y algunas de sus características. La clase Videojuego depende de esta clase porque sin ella no tendría sentido, ya que un videojuego necesita una plataforma donde jugarse. (Esta clase y sus métodos son abstractos, no se muestran correctamente en la imagen por un fallo del programa).

Métodos:

iniciar(): Este método dará inicio al juego en la plataforma deseada.

detener(): Este método detendrá el juego.

conectarRed(): Este método nos conectará a la red si es necesario para jugar.

desconectarRed(): Este método nos desconectará de la red.

Clase PC.

Descripción: Esta clase es una derivada de la interfaz Plataforma, siendo un tipo de plataforma.

Atributos:

modelo: Contiene el modelo del PC.

sistemaOperativo: Contiene el sistema operativo del PC.

procesador: Contiene el modelo del procesador del PC.

ram: Contiene la cantidad de RAM que tiene el PC.

Métodos:

getModelo(): Muestra por pantalla el modelo del PC.

verCaracterísticas(): Este método mostrará por pantalla las características del PC.

Clase Consola.

Descripción: Esta clase es una derivada de la interfaz Plataforma, siendo un tipo de plataforma.

Atributos:

modelo: Contiene el modelo de la consola, por ejemplo si es una PlayStation o si es una Xbox.

almacenamiento: Contiene la cantidad de almacenamiento que tiene la consola.

esPortatil: Este atributo booleano nos indicará si es portátil en caso de que la respuesta sea true y si la respuesta es false no es portátil.

Métodos:

getModelo(): Muestra por pantalla el modelo de la consola.

isEsPortatil(): Con este método se puede comprobar si la consola es portátil o no.

verConsola(): Muestra por pantalla todas las características de la consola.

sincronizarMando(): Este método conecta el mando a la consola para poder empezar a jugar.

Clase Movil.

Descripción: Esta clase es una derivada de la interfaz Plataforma, siendo un tipo de plataforma.

Atributos:

modelo: Indica el sistema operativo del dispositivo móvil.

bateria: Contiene la cantidad de batería restante que le queda al móvil.

notificaciones: Este atributo booleano contiene la configuración de las notificaciones del móvil.

Métodos:

getModelo(): Muestra por pantalla el modelo del móvil.

isNotificaciones(): Con este método podemos saber si las notificaciones están activadas o desactivadas.

verDetalles(): Este método mostrará por pantalla las características del móvil.

controlarNotificaciones(): Este método servirá para regular las notificaciones del juego en el dispositivo móvil.

Pruebas unitarias

Test pruebaGetPrecio()

Esta prueba verifica que el método `getPrecio()` de la clase `Videojuego` funcione correctamente.

En la prueba creamos un objeto `videojuego` con un precio asignado de 2, posteriormente llamamos al método para obtener el valor almacenado internamente y con el `assertEquals` comprobamos que el precio sea exactamente 2. El 0.01 es la tolerancia del error permitida, esto puede pasar por posibles diferencias de punto flotante.

Test pruebaGetPerfilJugador()

Esta prueba verifica que el método `getPerfilJugador(Jugador j)` de la clase `Videojuego` devuelva correctamente la información del jugador en formato de texto.

En la prueba primero creamos un objeto `Jugador` con la estructura de su constructor, luego creamos un objeto `Videojuego` asociándolo al jugador creado, procedemos a llamar al método para que devuelva un `String` con la información del jugador, definimos el texto esperado que debe devolver el método y se compara la salida real con la esperada, si son iguales el test pasa la prueba.

Test pruebaSetIdJugador()

Esta prueba comprueba que el método `setIdJugador()` de la clase `Videojuego` asigne correctamente un objeto `Jugador` al videojuego.

En la prueba creamos un objeto `Jugador` y un objeto `Videojuego` pero indicando null en el constructor para no asignar al jugador, invocamos al método `setIdJugador()` asignándole el jugador al videojuego y con `assertEquals` verificamos que el jugador que se ha asignado al videojuego es el mismo que se estableció.

Test pruebaAumentarNivel()

Esta prueba verifica que el método `aumentarNivel()` de la clase `Jugador` incremente correctamente un nivel del jugador.

En la prueba creamos un objeto `Jugador`, es importante asignarle un nivel, llamamos al método `aumentarNivel()` y con `assertEquals` verificamos que el nuevo nivel del jugador sea uno más que el que le hemos asignado nosotros.

Tests pruebaEsPortatil1() y pruebaEsPortatil2()

Estas dos pruebas verifican que el método `isEsPortatil()` de la clase `Consola` devuelve correctamente true o false si la consola es portátil o no.

En la prueba creamos un objeto `Consola`, indicando con true o false si es portátil o no y con `assertTrue` comprobamos que el método `isEsPortatil` devuelva true como se espera en el primer método y false como se espera en el segundo método.

Diagrama de Clase.

