**Contratos**

En el documento se van a mostrar los contratos de todos los metodos establecidos en el diseño. Para facilitar la lectura del documento y la identificacion de cada uno de los metodos, se presentan divididos por clase.

1. **Metodos de InfoParque**

* anadirAtraccion

public void anadirAtraccion(Atracccion atraccion)

Añade una nueva atraccion al parque.

*Precondiciones:*

* + La lista de atracciones esta creada.

*Post:*

* + Se añade la nueva atraccion a la lista.

*Parametros:*

* + atraccion – Instancia de la clase atraccion con todas sus caracteristicas y ademas clasificada en alguna sublcase, ya sea mecanica o cultural.

* anadirEspectaculo

public void anadirEspectaculo(Espectaculo espectaculo)

Añade un nuevo espectaculo al parque.

*Precondiciones:*

* + La lista de espectaculos esta creada.

*Post:*

* + Se añade el nuevo espectaculo a la lista.

*Parametros:*

* + espectaculo – Instancia de la clase espectaculo con todas las caracteristicas necesarias.
* eliminarAtraccion

public void eliminarAtraccion(Atraccion atraccion)

Elimina una atraccion del parque.

*Precondiciones:*

* + La lista de atracciones esta creada.

Post:

* + Se elimina la atraccion de la lista.

Parametros:

* + atraccion – Instancia de la clase atraccion con todas sus caracteristicas y ademas clasificada en alguna sublcase, ya sea mecanica o cultural.
* eliminarEspectaculo

public void eliminarEspectaculo(Espectaculo espectaculo)

Elimina un espectaculo del parque.

*Precondiciones:*

* + La lista de espectaculos esta creada.

*Post:*

* + Se elimina el espectaculo de la lista.

*Parametros:*

* + espectaculo – Instancia de la clase espectaculo con todas las caracteristicas necesarias.
* mostrarAtracciones

public List<Atraccion> mostrarAtracciones()

Muestra la lista de atracciones a los usuarios.

*Precondiciones:*

* + La lista de atracciones esta creada.

Post:

* + Muestra la lista de atracciones.
* mostrarEspectaculo

public List<Espectaculo> mostrarEspectaculos()

Mostrar la lista de espectaculos a los usuarios.

*Precondiciones:*

* + La lista de espectaculos esta creada.

*Post:*

* + Muesra la lista de espectaculos.
* existeAtraccion

public bool existeAtraccion(String aNombre)

Verifica si una atraccion ya existe dentro del parque.

*Precondiciones:*

* + La lista de atracciones esta creada.

Post:

* + Mediante un booleano se indica si la atraccion existe o no.

Parametros:

* + aNombre – Un string del nombre de la atraccion que se esta buscando.
* existeEspectaculo

public bool existeEspectaculo(String eNombre)

Verifica si un espectaculo existe dentro del parque.

*Precondiciones:*

* + La lista de espectaculos esta creada.

*Post:*

* + Mediante un booleano se indica si el espectaculo existe o no.

*Parametros:*

* + eNombre – Un string del nombre del espectaculo que se esta buscando.
* actualizarHorarios

public void actualizarHorarios(List<DateTime> horarios)

Actualiza los horarios en los que esta abierto el parque.

*Post:*

* + Se actualiza la lista de los horarios.

*Parametros:*

* + horarios – Una lista de 8 elementos, donde por orden de los dias de la semana(el octavo elemento es para los festivos), se muestra el intervalo de tiempo en el cual el parque esta abierto.
* mostrarHorarios

public List<DateTime> mostrarHorarios()

Muestra los horarios del parque.

*Precondiciones:*

* + La lista de horarios esta creada

*Post:*

* + Se muestra la lista de los horarios.

1. **Metodos de InfoUsuarios**

* existeUsuario

public bool existeUsuario(String uNombre)

Verifica si existe un usuario registrado con el mismo login dentro del parque.

*Precondiciones:*

* + La lista de usuarios esta creada.

*Post:*

* + Mediante un booleano se indica si el usuario existe o no.

*Parametros:*

* + uNombre – Un string del login del usuario que se esta buscando.
* anadirUsuario

public void anadirUsuario(String uLogin)

Añade un nuevo usuario a la lista de usuarios

*Precondiciones:*

* + La lista de usuarios esta creada.
  + Se verifico que el String que se recibe por parametro no existe.

*Post:*

* + Se añade el usuario a la lista de usuarios

*Parametros:*

* + uLogin – Un string del login del usuario que se va a añadir.
* eliminarUsuario

public void eliminarUsuario(String uLogin)

Elimina un usuario de la lista de usuarios del parque.

*Precondiciones:*

* + La lista de usuarios esta creada.

*Post:*

* + Se elimina el usuario de la lista de usuarios.

*Parametros:*

* + uLogin – Un string del login del usuario que se va a eliminar.

1. **Metodos de Usuario**

* cambioContraseña

public void cambiarContrasena()

Cambia la contraseña de el usuario que lo solicite.

*Precondiciones:*

* + La lista de usuarios esta creada.

*Post:*

* + Se cambia la contraseña del usuario.
* comprobarInicioSesion

public bool comprobarInicioSesion(String uLogin,String uContra)

Verifica que el usuario coincide con la contraseña establecida.

*Precondiciones:*

* + La lista de usuarios esta creada
  + El usuario existe y esta en la lista

*Post:*

* + Mediante un booleano se indica si la informacion coincide o no.

*Parametros:*

* + uLogin – Un string del login del usuario.
  + uContra – Un string con la clave que digita el usuario en consola.
* iniciarSesion

public bool iniciarSesion()

Inicia el proceso para que un usuario pueda iniciar sesion.

*Precondiciones:*

* + La lista de usuarios esta creada.

*Post:*

* + Mediante un booleano se indica si el inicio de sesion fue exitoso y puede acceder a la pagina.
* crearLogin

public bool crearLogin()

Este metodo se utiliza cuando un usuario nuevo quiere acceder a la pagina del parque, mediante este se crea un login y se asigna una contraseña.

*Precondiciones:*

* + La lista de usuarios esta creada

*Post:*

* + Crea un nuevo usuario y lo añade a la lista de usuarios.

*Parametros:*

* + uLogin – Un string del login del usuario.
  + uContra – Un string con la clave que digita el usuario en consola.

1. **Métodos de Atraccion**

* verificarCondicionesOperacion

public void verificarCondicionesOperacion(Atraccion atraccion)  
Comprueba que una atracción reúne todas las condiciones necesarias (clima, capacidad, exclusividad) para operar.

*Precondiciones:*

* La instancia atraccion no es nula.
* Sus atributos (cupoMaximo, restriccionClima, etc.) están inicializados.

*Postcondiciones:*

* Se evalúan y registran las condiciones de operación.

Parámetros:

* atraccion – Instancia de la clase Atraccion a verificar.
* actualizarAtraccion

public void actualizarAtraccion(Atraccion atraccion)  
Actualiza los datos de una atracción existente en el sistema.

*Precondiciones:*

* La instancia atraccion no es nula.
* La atracción ya existe en la lista del parque.

*Postcondiciones:*

* Los datos de la atracción se actualizan según los atributos de la instancia.

*Parámetros:*

* atraccion – Instancia de la clase Atraccion con nuevos valores.

1. **Métodos de Empleado**

* crearEmpleado

public boolean crearEmpleado()  
Crea y registra un nuevo empleado en el sistema.

*Precondiciones:*

* Se dispone de los datos necesarios (nombre, capacitaciones).

*Postcondiciones:*

* Se añade el nuevo Empleado a la lista de empleados.
* Retorna **true** si la creación fue exitosa, **false** en caso contrario.

1. **Métodos de TiqueteTemporada**

* verificarFechas

public boolean verificarFechas(TiqueteTemporada tiqueteTemporada)  
Valida que las fechas del ticket de temporada estén dentro del rango permitido.

*Precondiciones:*

* El objeto tiqueteTemporada no es nulo.
* Su atributo temporada y Limite están inicializados.

*Postcondiciones:*

* Retorna **true** si las fechas son válidas, **false** en caso contrario.

Parámetros:

* tiqueteTemporada – Instancia de TiqueteTemporada a validar.

1. **Métodos de TiqueteIndividual**

* actualizarUso

public void actualizarUso(String ticketId, TiqueteIndividual tiqueteIndividual)  
Marca un ticket individual como usado.

*Precondiciones:*

* ticketId no es nulo ni vacío.
* La instancia tiqueteIndividual existe y coincide con ticketId.

*Postcondiciones:*

* El atributo uso del ticket pasa a **true**.

*Parámetros:*

* ticketId – Identificador del ticket.
* tiqueteIndividual – Instancia de TiqueteIndividual a actualizar.
* asignarAtraccion

public void asignarAtraccion(Atraccion atraccion)  
Asigna una atracción a un ticket individual.

*Precondiciones:*

* atraccion no es nula.
* El ticket no ha sido usado.

*Postcondiciones:*

* El ticket queda vinculado a la atracción indicada.

*Parámetros:*

* atraccion – Instancia de Atraccion a asignar.

1. **Métodos de TiqueteSinFecha**

* actualizarUso

public void actualizarUso(String ticketId, TiqueteSinFecha tiquete)  
Marca un ticket sin fecha como utilizado.

*Precondiciones:*

* ticketId no es nulo ni vacío.
* tiquete coincide con ticketId.

*Postcondiciones:*

* El atributo uso se establece en **true**.

*Parámetros:*

* ticketId – Identificador del ticket.
* tiquete – Instancia de TiqueteSinFecha a actualizar.
* asignarExclusividad

public void asignarExclusividad(String exclusividad, String ticketId,Tiquete tiquete)  
Define la exclusividad de un ticket sin fecha.

*Precondiciones:*

* exclusividad no es nulo ni vacío.
* ticketId no es nulo ni vacio
* tiquete coincide con el ticketId

*Postcondiciones:*

* El ticket queda asignado a la modalidad indicada.

*Parámetros:*

* exclusividad – Cadena que describe el tipo de exclusividad.
* ticketId – identificador del ticket
* tiquete – Instancia de TiqueteSinFecha a actualizar.

1. **Métodos de Tiquete (base)**

* compraOnline

public void compraOnline()  
Procesa la compra de un ticket a través de la plataforma online.

*Precondiciones:*

* Lista de tiquetes existe
* Lista de atracciones existe

*Postcondiciones:*

* Se crea el tiquete que escoja el usuario y se asigna al mismo.
* compraTaquilla

public void compraTaquilla(LugarServicio puntoVenta)  
Procesa la compra de un ticket en taquilla.

*Precondiciones:*

* puntoVenta no es nulo ni vacío.
* Lista de tiquetes existe
* Lista de atracciones existe

Postcondiciones:

* Se genera el ticket en el punto de venta indicado y se asocia al usuario.

Parámetros:

* puntoVenta –LugarServicio con tipo asociado “Taquilla”
* asignarFastPass()

public void asignarFastpass(Tiquete tiquete)  
Vincula un fastpass al ticket.

*Precondiciones:*

* El tiquete ingresado no es vacio ni nulo

*Postcondiciones:*

* El ticket queda marcado con fastpass.

Parámetros:

* Tiquete – Una instancia de la clase tiquete a la cual se le compro fastpass.

1. **Métodos de Turno**

* asginarLugarTrabajo

public void asignarLugarTrabajo(LugarServicio lugarServicio)  
Asigna un lugar de trabajo a un turno.

*Precondiciones:*

* lugarServicio no es nulo.
* El turno no tiene un lugar ya asignado

*Postcondiciones:*

* El turno queda vinculado al lugar indicado.

*Parámetros:*

* lugarServicio – Instancia de LugarServicio.
* asignarTurnoEmpleado

public void asignarTurnoEmpleado(Empleado empleado, Turno turno)

Asigna un turno especifico a un empleado en el sistema.

*Precondiciones:*

* La instancia empleado no es nula y esta registrada en el sistema
* El turno no es nulo y no esta asignado a otro empleado

*Postcondiciones:*

* El turno queda vinculado al empleado
* Se actualiza la lista de turnos del empleado para incluir el turno.

*Parámetros:*

* empleado – Instancia de la clase empleado
* turno – Instancia de la clase turno
* verificarCondicionesEmpleado

public bool verificarCondicionesEmpleado(Empleado empleado)  
Verifica que un empleado cumple con los requisitos para un turno.

*Precondiciones:*

* nombreEmpleado no es nulo ni vacío.
* empleado existe en el sistema

*Postcondiciones:*

* Retorna **true** si cumple condiciones, **false** en caso contrario.

*Parámetros:*

* empleado – Instancia de Empleado.
* verificarCondicionesLugar

public boolean verificarCondicionesLugar(LugarServicio lugar)  
Comprueban que un lugar tenga suficientes empleados para operar.

*Precondiciones:*

* lugar no es nulo

*Postcondiciones:*

* Retorna **true** si se puede operar, **false** en caso contrario.

*Parámetros:*

* lugar – Instancia de LugarServicio.

1. **Métodos de FastPass**

* actualizarUso

public void actualizarUso(Tiquete tiquete)  
Marca el fastpass como utilizado.

*Precondiciones:*

* El objeto FastPass existe y no ha sido usado.

Postcondiciones:

* El atributo uso se establece en **true**.

Parámetros:

* tiquete – Instancia de la clase Tiquete

1. **Métodos de Espectaculo**

* actualizarEspectaculo

public void actualizarEspectaculo(Espectaculo espectaculo)  
Actualiza los datos de un espectáculo existente.

*Precondiciones:*

* espectaculo no es nulo.
* El espectáculo existe en la lista.

*Postcondiciones:*

* Se actualizan los atributos (horario, ubicación, clima, disponibilidad)

*Parámetros:*

* espectaculo – Instancia de Espectaculo.

1. **Métodos de InfoTiquetes**

* mostrarTiquetes

public List<Tiquete> mostrarTiquetes()  
Devuelve la lista de todos los tickets disponibles.

*Precondiciones:*

* La colección de tiquetes está inicializada.

*Postcondiciones:*

* Retorna la lista completa de instancias Tiquete.

Con la informacion mostrada se abarcan todos los metodos para los cuales es necesario realizar un contrato, los constructores o los setters y getters no es necesario realizarles un contrato ya que estos por definicion se sabe como implementarlos.