**Documentación del Proyecto: F1 Manager 2024 (Consola)**

**Integrantes: Freddy Bautista, Javier Esquivel, Sebastian Viloria**

**Descripción General:**

Este proyecto es una aplicación de consola desarrollada en Java que permite gestionar y consultar información detallada sobre la temporada 2024 de Fórmula 1. Carga datos estáticos (hardcodeados) al inicio y ofrece un menú para visualizar información sobre equipos, pilotos, circuitos y los resultados de cada carrera de la temporada.

**Estructura del Proyecto:**

El proyecto se organiza principalmente en:

1. **Paquete model:** Contiene las clases que representan las entidades del dominio (Equipos, Pilotos, Carreras, etc.).
2. **Paquete data:** Contiene la clase responsable de cargar los datos iniciales.
3. **Clase Main (Paquete por defecto):** Contiene la lógica de la interfaz de usuario (menú de consola) y el punto de entrada de la aplicación.
4. **Extra:** En el directorio “Docs” que esta en la raíz del proyecto se encuentra la documentación del mismo con los diagramas UML de cada clase, el diagrama general, las imágenes e incluso este mismo archivo Docx.

**Paquete: model**

Este paquete contiene las clases que definen la estructura de los datos del dominio de la F1.

**1. Clase Competidor (Abstracta)**

* **Propósito:** Clase base abstracta para representar entidades que compiten en el campeonato (Equipos y Pilotos). Agrupa atributos comunes y permite aplicar polimorfismo.
* **Atributos:**
  + nombreCompleto (String, final): El nombre completo del competidor (equipo o piloto). Es inmutable una vez creado.
  + campeonatosGanados (int, final): El número total de campeonatos mundiales ganados históricamente por el competidor hasta el inicio o fin de la temporada de referencia. Es inmutable.
  + puntos2024 (double, final): Los puntos totales obtenidos por el competidor específicamente en la temporada 2024. Es inmutable.
* **Métodos:**
  + Competidor(String nombreCompleto, int campeonatosGanados, double puntos2024): Constructor para inicializar los atributos comunes.
  + getNombreCompleto(): Devuelve el nombre completo (String).
  + getCampeonatosGanados(): Devuelve el número de campeonatos ganados (int).
  + getPuntos2024(): Devuelve los puntos obtenidos en la temporada 2024 (double).

**2. Clase Equipo**

* **Propósito:** Representa a un equipo (escudería) de Fórmula 1 participante en la temporada 2024. Hereda de Competidor.
* **Atributos:**
  + id (int, final): Identificador numérico único para el equipo.
  + directorGeneral (String, final): Nombre del Jefe de Equipo o Director General principal durante 2024.
  + paisOrigen (String, final): País donde el equipo tiene su base principal o licencia.
  + proveedorMotor (String, final): Fabricante que suministró los motores al equipo en 2024.
  + carrerasGanadas2024 (Integer, final): Número de Grandes Premios ganados por los pilotos de este equipo durante la temporada 2024. Es Integer para permitir null si no hubo victorias (aunque se carga como 0 en ese caso).
  + ultimaVictoria (LocalDate, final): Fecha de la última victoria conocida del equipo en un Gran Premio (hasta el final de 2024). Puede ser null si no hay datos o nunca ha ganado.
  + pilotos (List&lt;Piloto>, final): Una lista que contiene las instancias de los objetos Piloto asociados a este equipo para la temporada 2024. Esta lista es poblada externamente (por CargadorDatos).
* **Métodos:**
  + Equipo(int id, String nombreEquipo, int campeonatosGanados, double puntos2024, String directorGeneral, String paisOrigen, String proveedorMotor, Integer carrerasGanadas2024, LocalDate ultimaVictoria): Constructor para inicializar un objeto Equipo. Llama al constructor de Competidor.
  + getId(): Devuelve el ID del equipo (int).
  + getDirectorGeneral(): Devuelve el nombre del director (String).
  + getPaisOrigen(): Devuelve el país base (String).
  + getProveedorMotor(): Devuelve el proveedor del motor (String).
  + getCarrerasGanadas2024(): Devuelve el número de carreras ganadas en 2024 (Integer).
  + getUltimaVictoria(): Devuelve la fecha de la última victoria (LocalDate).
  + agregarPiloto(Piloto piloto): Añade un objeto Piloto a la lista interna pilotos. Usado por CargadorDatos para establecer la relación.
  + getPilotos(): Devuelve la lista de pilotos (List<Piloto>) asociados a este equipo.

**3. Clase Piloto**

* **Propósito:** Representa a un piloto de Fórmula 1 que compitió en la temporada 2024. Hereda de Competidor.
* **Atributos:**
  + id (int, final): Identificador numérico único para el piloto.
  + equipo (Equipo, final): Referencia al objeto Equipo al que pertenece el piloto en 2024 (o su equipo principal en caso de sustitutos).
  + numero (int, final): Número permanente del coche usado por el piloto.
  + abreviatura (String, final): Código de 3 letras usado para identificar al piloto (ej. "VER", "HAM", "ALO").
  + carrerasDisputadas (int, final): Número total aproximado de Grandes Premios en los que ha participado el piloto en toda su carrera hasta el final de 2024.
  + carrerasGanadas (Integer, final): Número total de victorias en Grandes Premios en toda su carrera hasta el final de 2024.
  + podios (Integer, final): Número total de podios (1°, 2° o 3°) en toda su carrera hasta el final de 2024.
  + poles (Integer, final): Número total de pole positions en toda su carrera hasta el final de 2024.
  + vueltasRapidas (Integer, final): Número total de vueltas rápidas en carrera en toda su carrera hasta el final de 2024.
  + posicionesPorCarrera (Map&lt;String, String>, final): Mapa (posiblemente obsoleto o redundante si Carrera tiene los datos) que originalmente contenía la posición del piloto por nombre de carrera.
  + nacionalidad (String, final): Nacionalidad del piloto.
  + edad (int, final): Edad del piloto en el momento en que se cargaron los datos.
  + temporadaActual (TemporadaActual, final): Referencia a un objeto TemporadaActual que contiene datos resumidos del estado del piloto al final de la temporada 2024.
* **Métodos:**
  + Piloto(int id, String nombreCompleto, int campeonatosGanados, double puntos2024, Equipo equipo, int numero, String abreviatura, int carrerasDisputadas, Integer carrerasGanadas, Integer podios, Integer poles, Integer vueltasRapidas, Map<String, String> posicionesPorCarrera, String nacionalidad, int edad, TemporadaActual temporadaActual): Constructor para inicializar un objeto Piloto. Llama al constructor de Competidor.
  + getId(): Devuelve el ID del piloto (int).
  + getEquipo(): Devuelve el objeto Equipo asociado (Equipo).
  + getNumero(): Devuelve el número del piloto (int).
  + getAbreviatura(): Devuelve la abreviatura del piloto (String).
  + getCarrerasDisputadas(): Devuelve el número de GPs disputados en su carrera (int).
  + getCarrerasGanadas(): Devuelve el número de victorias en su carrera (Integer).
  + getPodios(): Devuelve el número de podios en su carrera (Integer).
  + getPoles(): Devuelve el número de poles en su carrera (Integer).
  + getVueltasRapidas(): Devuelve el número de vueltas rápidas en su carrera (Integer).
  + getPosicionesPorCarrera(): Devuelve el mapa de posiciones por carrera (Map<String, String>).
  + getNacionalidad(): Devuelve la nacionalidad (String).
  + getEdad(): Devuelve la edad (int).
  + getTemporadaActual(): Devuelve el objeto TemporadaActual asociado (TemporadaActual).
  + toString(): Devuelve una representación simple en String del piloto (nombre, equipo, puntos 2024).

**4. Clase TemporadaActual**

* **Propósito:** Encapsula datos específicos del estado o resultado de un piloto al finalizar la temporada 2024.
* **Atributos:**
  + posicionCampeonato (int, final): Posición final del piloto en el Campeonato Mundial de Pilotos 2024.
  + carrerasCompletadas (int, final): Número de carreras en las que el piloto participó durante la temporada 2024.
  + carrerasPendientes (int, final): Número de carreras que faltaban en la temporada (probablemente 0 si los datos son finales).
  + proximaCarrera (String, final): Nombre de la siguiente carrera o de la última carrera relevante en el contexto de los datos cargados.
  + fechaProximaCarrera (LocalDate, final): Fecha de esa próxima o última carrera relevante.
* **Métodos:**
  + TemporadaActual(int posicionCampeonato, int carrerasCompletadas, int carrerasPendientes, String proximaCarrera, LocalDate fechaProximaCarrera): Constructor para inicializar el objeto.
  + getPosicionCampeonato(): Devuelve la posición final en el campeonato (int).
  + getCarrerasCompletadas(): Devuelve el número de carreras completadas por el piloto en 2024 (int).
  + getCarrerasPendientes(): Devuelve el número de carreras pendientes (int).
  + getProximaCarrera(): Devuelve el nombre de la próxima/última carrera (String).
  + getFechaProximaCarrera(): Devuelve la fecha de la próxima/última carrera (LocalDate).

**5. Clase Circuito**

* **Propósito:** Representa una pista o circuito donde se celebra un Gran Premio de F1.
* **Atributos:**
  + id (int, final): Identificador numérico único para el circuito en el calendario 2024.
  + nombre (String, final): Nombre oficial del circuito (ej. "Silverstone Circuit").
  + pais (String, final): País donde se ubica el circuito.
  + fechaCarreraPrincipal (LocalDate, final): Fecha en que se celebró la carrera principal (Gran Premio) en este circuito en 2024.
  + fechaCarreraSprint (LocalDate, final): Fecha en que se celebró la carrera Sprint en este circuito en 2024. Es null si no hubo Sprint en ese circuito.
* **Métodos:**
  + Circuito(int id, String nombre, String pais, LocalDate fechaCarreraPrincipal, LocalDate fechaCarreraSprint): Constructor para inicializar el objeto.
  + getId(): Devuelve el ID del circuito (int).
  + getNombre(): Devuelve el nombre del circuito (String).
  + getPais(): Devuelve el país (String).
  + getFechaCarreraPrincipal(): Devuelve la fecha de la carrera principal (LocalDate).
  + getFechaCarreraSprint(): Devuelve la fecha de la carrera sprint (LocalDate, puede ser null).

**6. Clase Carrera**

* **Propósito:** Representa un evento de carrera específico (Gran Premio principal o Sprint) dentro de la temporada. Contiene los resultados detallados de ese evento.
* **Atributos:**
  + nombre (String, final): Nombre del evento (ej. "GP de Bahrein", "Sprint de China").
  + circuito (Circuito, final): Referencia al objeto Circuito donde tuvo lugar la carrera.
  + fecha (LocalDate, final): Fecha exacta en que se realizó este evento de carrera.
  + tipo (String, final): Indica si es la carrera "Principal" o un "Sprint".
  + posicionesPilotos (Map&lt;Piloto, String>, final): Mapa que asocia cada objeto Piloto participante con su resultado final en esta carrera (posición como "1", "2", o estado como "DNF", "DSQ").
  + puntosPilotos (Map&lt;Piloto, Integer>, final): Mapa que asocia cada objeto Piloto participante con los puntos exactos que obtuvo *en esta carrera específica*.
  + puntosEquipos (Map&lt;Equipo, Integer>, final): Mapa que asocia cada objeto Equipo con el total de puntos sumados por sus pilotos *en esta carrera específica*.
* **Métodos:**
  + Carrera(String nombre, Circuito circuito, LocalDate fecha, String tipo, Map<Piloto, String> posicionesPilotos, Map<Piloto, Integer> puntosPilotos, Map<Equipo, Integer> puntosEquipos): Constructor para inicializar el objeto.
  + getNombre(): Devuelve el nombre de la carrera (String).
  + getCircuito(): Devuelve el objeto Circuito asociado (Circuito).
  + getFecha(): Devuelve la fecha de la carrera (LocalDate).
  + getTipo(): Devuelve el tipo de carrera (String).
  + getPosicionesPilotos(): Devuelve el mapa de posiciones de pilotos (Map<Piloto, String>).
  + getPuntosPilotos(): Devuelve el mapa de puntos de pilotos (Map<Piloto, Integer>).
  + getPuntosEquipos(): Devuelve el mapa de puntos de equipos (Map<Equipo, Integer>).

**Paquete: data**

Este paquete contiene las clases encargadas de la creación y carga de los datos.

**1. Clase CargadorDatos**

* **Propósito:** Clase de utilidad (con métodos estáticos) responsable de instanciar y llenar todos los objetos del modelo (Equipo, Piloto, Circuito, Carrera) con los datos estáticos y hardcodeados correspondientes a la temporada completa de F1 2024. Es la fuente única de datos para la aplicación.
* **Atributos:** Ninguno (solo métodos estáticos).
* **Métodos:**
  + cargarEquipos() -> List<Equipo> (static): Crea instancias de los 10 equipos de F1 de 2024, les asigna sus datos finales (puntos 2024, director, motor, país, etc.) y los devuelve en una lista.
  + cargarCircuitos() -> List<Circuito> (static): Crea instancias de los 24 circuitos del calendario 2024 con sus nombres, países y fechas de carrera (principal y sprint si aplica), y los devuelve en una lista.
  + cargarPilotos() -> List<Piloto> (static): Crea instancias de los 23 pilotos que compitieron en 2024. Asigna a cada uno su Equipo correspondiente (obtenido de cargarEquipos), sus datos personales (número, nacionalidad, edad), sus estadísticas de *carrera* (victorias, podios, etc. hasta fin de 2024), sus puntos *finales de la temporada 2024*, y un objeto TemporadaActual con el resumen final. **Importante:** Después de crear la lista de pilotos, este método realiza la **vinculación** llamando a equipo.agregarPiloto(piloto) para añadir cada piloto a la lista interna de su respectivo objeto Equipo. Devuelve la lista completa de pilotos.
  + cargarCarreras(List<Piloto> pilotos, List<Equipo> equipos, List<Circuito> circuitos) -> List<Carrera> (static): Recibe las listas de pilotos, equipos y circuitos ya cargadas. Para cada uno de los 24 Grandes Premios de 2024, crea un objeto Carrera. Utiliza mapas definidos internamente (Map.ofEntries con IDs de piloto) y una función helper (TriFunction lambda) para transformar esos datos crudos en los mapas requeridos por el constructor de Carrera (Map<Piloto, String>, Map<Piloto, Integer>, Map<Equipo, Integer>) llenos con los **resultados reales y detallados** de esa carrera específica. Devuelve la lista completa de las 24 carreras con sus resultados.
  + TriFunction (private interface): Interfaz funcional interna usada como plantilla por la expresión lambda crearCarrera para estandarizar la creación de objetos Carrera a partir de los datos de resultados.

**Clase Main (Paquete por defecto)**

* **Propósito:** Punto de entrada principal de la aplicación (main method). Gestiona la interacción con el usuario a través de un menú de consola. Orquesta la carga de datos y la visualización de la información solicitada por el usuario.
* **Atributos:**
  + DATE\_FORMATTER (static final DateTimeFormatter): Objeto para formatear objetos LocalDate a un patrón legible (dd/MM/yyyy) en la salida por consola.
* **Métodos:**
  + main(String[] args) (static): Método principal que ejecuta la JVM. Simplemente llama a mostrarMenu() para iniciar la aplicación.
  + mostrarMenu() (static):
    1. Llama a los métodos de CargadorDatos para obtener las listas completas de equipos, pilotos, circuitos y carreras al inicio.
    2. Crea un objeto Scanner para leer la entrada del usuario.
    3. Entra en un bucle do-while que se repite hasta que el usuario elige la opción 0 (Salir).
    4. Dentro del bucle:
       - Imprime las opciones del menú.
       - Lee la opción numérica del usuario (con manejo básico de errores de entrada).
       - Limpia el buffer del scanner.
       - Usa una estructura switch para llamar al método correspondiente a la opción elegida.
    5. Al salir del bucle (opción 0), cierra el Scanner.
  + mostrarPosicionesPilotosEquipos(List<Carrera> carreras) (static): Implementa la opción 1 del menú. Itera sobre la lista de Carrera. Para cada una, imprime el nombre/lugar/fecha y luego lista las posiciones finales de los pilotos (ordenadas correctamente por un comparador personalizado) y los puntos obtenidos por cada equipo *en esa carrera*.
  + mostrarPuntosPorCarrera(List<Carrera> carreras) (static): Implementa la opción 2 del menú (nombre ajustado para claridad). Itera sobre la lista de Carrera. Para cada una, imprime el nombre/fecha y luego lista los puntos obtenidos por cada piloto y por cada equipo *exclusivamente en esa carrera*, ordenados de mayor a menor puntuación.
  + mostrarInfoEquipo(List<Equipo> equipos, Scanner scanner) (static): Implementa la opción 3. Pide al usuario un ID o nombre de equipo. Busca el equipo en la lista equipos. Si lo encuentra, imprime sus detalles: ID, nombre, director, país, motor, puntos 2024, campeonatos, carreras ganadas 2024, última victoria y, crucialmente, la **lista de pilotos asociados** obtenida mediante equipoEncontrado.getPilotos().
  + mostrarInfoPiloto(List<Piloto> pilotos, Scanner scanner) (static): Implementa la opción 4. Pide al usuario un ID, nombre o abreviatura de piloto. Busca el piloto en la lista pilotos. Si lo encuentra, imprime sus detalles: ID, nombre, abreviatura, número, equipo, nacionalidad, edad, **puntos 2024**, campeonatos, y sus **estadísticas históricas de carrera** (GPs, victorias, podios, poles, vueltas rápidas). También muestra la posición final en el campeonato 2024 desde TemporadaActual.
  + mostrarInfoCircuito(List<Circuito> circuitos, Scanner scanner) (static): Implementa la opción 5. Pide al usuario un ID o nombre de circuito. Busca el circuito en la lista circuitos (permite buscar por ambos). Si lo encuentra, imprime sus detalles: ID, nombre, país y fechas de carrera principal y sprint (si aplica) de 2024.
  + listarPilotos(List<Piloto> pilotos) (static): Implementa la opción 6. Recorre la lista pilotos e imprime una línea por cada uno con su ID, nombre, abreviatura y nombre del equipo. Ordena por ID.
  + listarEquipos(List<Equipo> equipos) (static): Implementa la opción 7. Recorre la lista equipos e imprime una línea por cada uno con su ID, nombre, director y país. Ordena por ID.
  + listarCircuitos(List<Circuito> circuitos) (static): Implementa la opción 8. Recorre la lista circuitos e imprime una línea por cada uno con su ID, nombre, país y fecha del GP principal 2024. Ordena por ID.

## 1. Comparación de Pilotos

### 1.1 Menú de Comparación

El sistema ofrece dos tipos de comparación para pilotos:  
- Comparación Simple: Muestra una comparación directa de puntos  
- Comparación Visual: Muestra una comparación gráfica detallada

### 1.2 Comparación Simple

Atributos mostrados:  
- Diferencia de Puntos: Muestra cuántos puntos más tiene un piloto sobre otro  
- Tabla Comparativa:  
 - Nombre del piloto  
 - Puntos totales en la temporada 2024

Formato de visualización:  
╔════════════════════════════════════════════════════════════╗  
║ 📊 COMPARACIÓN DE PILOTOS ║  
╠════════════════════════════════════════════════════════════╣  
║ [Nombre Piloto 1] tiene [X] puntos más que [Nombre Piloto 2]║  
╟────────────────────────────────────────────────────────────╢  
║ PILOTO │ PUNTOS 2024 ║  
╟────────────────────────────────────────────────────────────╢  
║ [Nombre Piloto 1] │ [Puntos] ║  
║ [Nombre Piloto 2] │ [Puntos] ║  
╚════════════════════════════════════════════════════════════╝

### 1.3 Comparación Visual

Muestra una comparación gráfica detallada de las estadísticas de los pilotos, incluyendo:  
- Victorias  
- Podios  
- Poles  
- Vueltas rápidas  
- Puntos totales

## 2. Comparación de Equipos

### 2.1 Menú de Comparación

El sistema ofrece dos tipos de comparación para equipos:  
- Comparación Simple: Muestra una comparación directa de puntos  
- Comparación Visual: Muestra una comparación gráfica detallada

### 2.2 Comparación Simple

Atributos mostrados:  
- Diferencia de Puntos: Muestra cuántos puntos más tiene un equipo sobre otro  
- Tabla Comparativa:  
 - Nombre del equipo  
 - Puntos totales en la temporada 2024

Formato de visualización:  
╔════════════════════════════════════════════════════════════╗  
║ 📊 COMPARACIÓN DE EQUIPOS ║  
╠════════════════════════════════════════════════════════════╣  
║ [Nombre Equipo 1] tiene [X] puntos más que [Nombre Equipo 2]║  
╟────────────────────────────────────────────────────────────╢  
║ EQUIPO │ PUNTOS 2024 ║  
╟────────────────────────────────────────────────────────────╢  
║ [Nombre Equipo 1] │ [Puntos] ║  
║ [Nombre Equipo 2] │ [Puntos] ║  
╚════════════════════════════════════════════════════════════╝

### 2.3 Comparación Visual

Muestra una comparación gráfica detallada de las estadísticas de los equipos, incluyendo:  
- Puntos totales  
- Carreras ganadas  
- Última victoria  
- Campeonatos históricos

## 3. Navegación

### 3.1 Acceso a las Comparaciones

- Comparación de Equipos: Opción 9 del menú principal  
- Comparación de Pilotos: Opción 10 del menú principal

### 3.2 Submenú de Comparación

Al seleccionar cualquiera de las opciones de comparación, se muestra un submenú con:  
1. Comparar posiciones y puntos  
2. Comparación visual  
3. Salir

### 3.3 Selección de Elementos

- Se muestra una lista numerada de todos los elementos disponibles  
- El usuario debe ingresar el número correspondiente a cada elemento que desea comparar  
- El sistema valida que las selecciones sean válidas

## 4. Manejo de Errores

### 4.1 Validaciones

- Verifica que existan al menos dos elementos para comparar  
- Valida que las selecciones de elementos sean números válidos  
- Verifica que las opciones del submenú sean válidas

### 4.2 Mensajes de Error

- "Se necesitan al menos dos [elementos] para comparar"  
- "Selección inválida"  
- "Opción inválida. Intente de nuevo"

## 5. Características Técnicas

### 5.1 Métodos Principales

- compararPilotos(List<Piloto> pilotos)  
- compararEquipos(List<Equipo> equipos)  
- mostrarComparacionSimplePilotos(Piloto piloto1, Piloto piloto2)  
- mostrarComparacionSimpleEquipos(Equipo equipo1, Equipo equipo2)

### 5.2 Formateo

- Utiliza caracteres Unicode para crear bordes y separadores  
- Implementa alineación de texto para mantener la consistencia visual  
- Maneja espacios dinámicos para adaptarse a diferentes longitudes de texto

### 5.3 Interacción con el Usuario

- Utiliza Scanner para la entrada de datos  
- Implementa limpieza del buffer después de cada entrada  
- Permite múltiples comparaciones en la misma sesión