F1 Manager

Didier Esteban Escamilla Solano

Greivin Arias Granados

Jared Gabriel Rosales Vargas

Daniel Josué Gómez Mora

Universidad Fidélitas

SC-202 introducción a la Programación

MSc. Andrés Vargas Rivera

Agosto, 2025

Table of Contents

Avance 1

Este documento explica cómo funciona el sistema de generación automática de identificadores (IDs) dentro del programa F1 Manager, y describe el diseño general del sistema, con la responsabilidad de cada clase y un diagrama que muestra su relación, hasta el avance del día 7/7/2025 por el grupo número 3.

Identificación de las clases principales y su responsabilidad

Corredor

Representa a un piloto del F1. Contiene información personal, habilidades, experiencia e ID del equipo. Su ID se genera automáticamente.

Equipo

Representa una escudería (Equipo). Almacena hasta dos corredores y designa a uno como principal. También tiene un ID automático.

F1_Manager

Registra equipos y corredores, asigna principales, simula carreras y muestra los datos.

Lógica de Generación de IDs Automáticos

En este sistema, tanto equipos como corredores necesitan un identificador único para poder diferenciarlos fácilmente. Para evitar errores o duplicados, los IDs se generan automáticamente usando una variable estática llamada contador.

En la clase Corredor:

this.id = String.format("DR-%05d", contador++);

Esto genera un ID como DR-00001, DR-00002, etc.

En la clase Equipo:

this.id = String.format("EQP-%05d", contador++);

Esto genera un ID como EQP-00001, EQP-00002, etc.

El uso de String.format permite mantener el formato consistente con ceros a la izquierda, y contador++ asegura que cada nuevo objeto tenga un número único.

Clases principales y su responsabilidad

Clase Corredor

- Representa un piloto de carreras.
- Atributos importantes: nombre, número, habilidad, experiencia, ID de equipo, y su ID único.
- El ID se asigna automáticamente al crearlo.
- También se asegura que cada corredor tenga un número diferente (por medio del conjunto numerosUsados en F1_Manager).

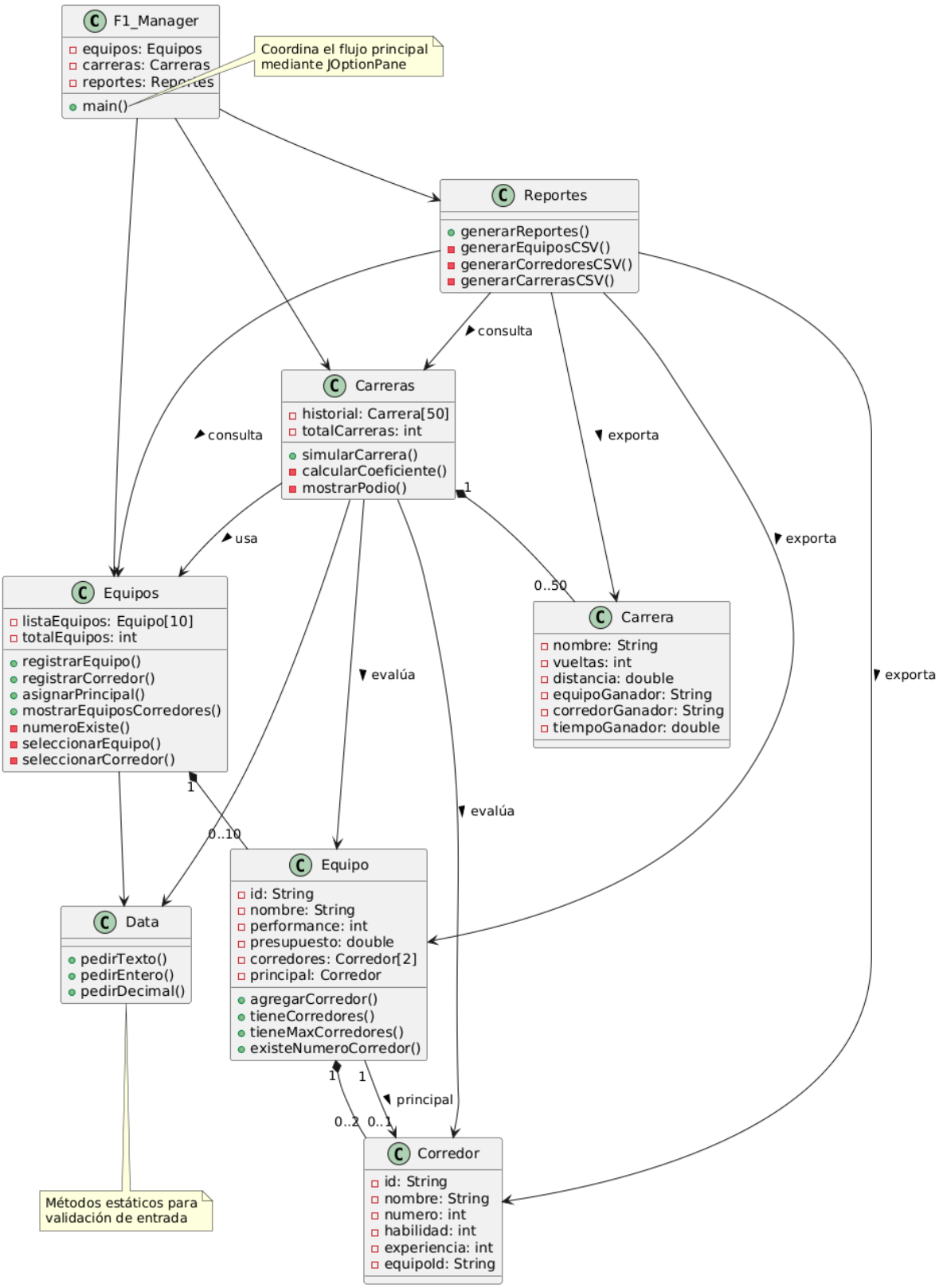
Clase Equipo

- Representa un equipo de F1.
- Puede tener hasta 2 corredores y uno de ellos puede ser designado como principal.
- Tiene su propio ID único automático, nombre, nivel de performance y presupuesto.
- Proporciona funciones para agregar corredores y verificar si hay espacio disponible.

Clase F1_Manager

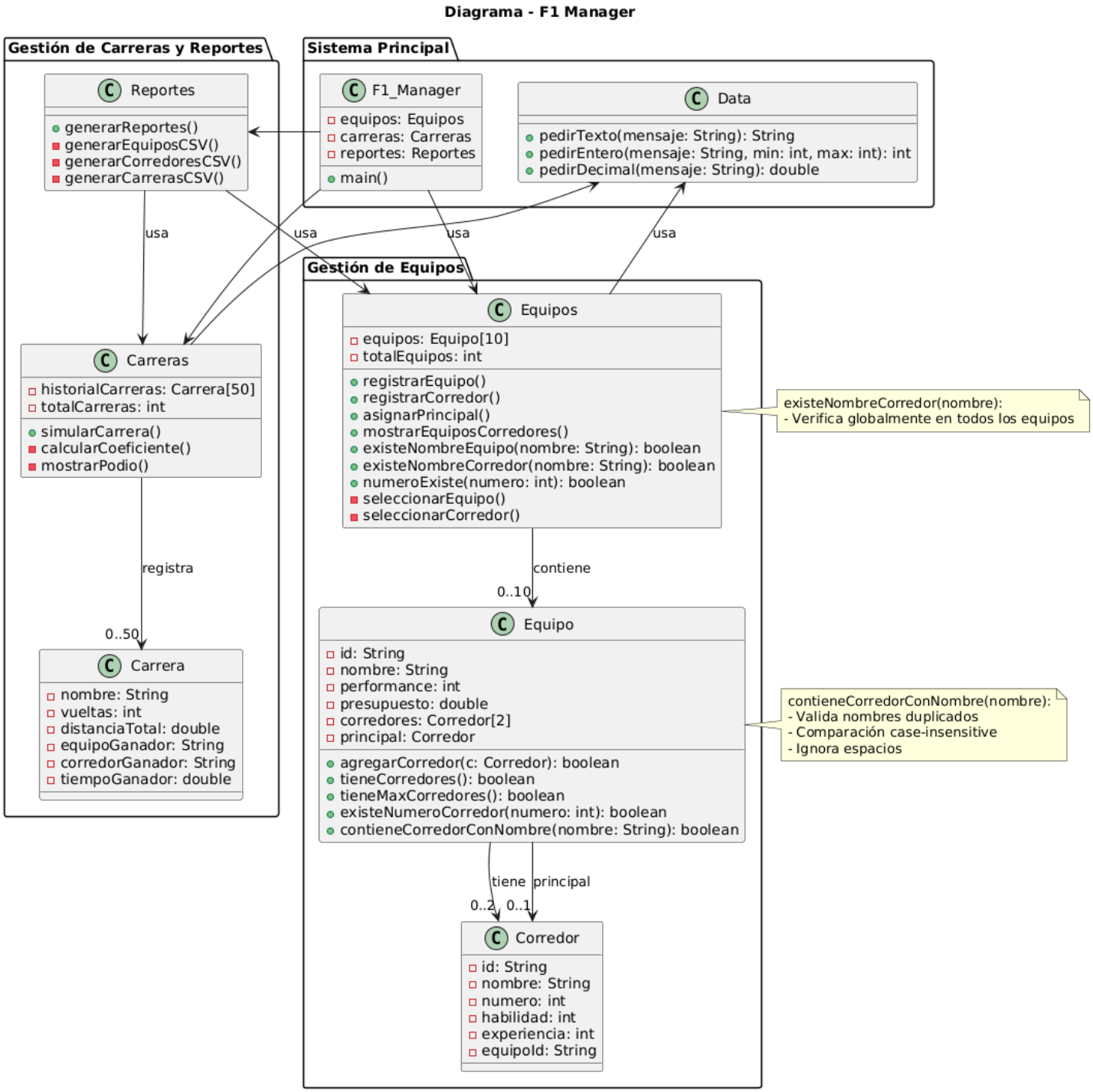
- Es la clase principal del programa.
- Controla todo el flujo: registrar equipos, registrar corredores, asignar principal, simular carreras y mostrar la información.
- Administra una lista de corredores y un arreglo de equipos.
- Se encarga de que los números de corredores no se repitan y de organizar la lógica del programa.

Diagrama


F1_Manager
equipos: Equipos
carreras: Carreras
reportes: Reportes
main()
Coordina el flujo principal mediante JOptionPane
Reportes
generarReportes()
generarEquiposCSV()
generarCorredoresCSV()
generarCarrerasCSV()
consulta
consulta
exporta
exporta
exporta
Carreras
historial: Carrera[50]
totalCarreras: int
simularCarrera()
calcularCoeficiente()
mostrarPodio()
1
usa
0..50
Equipos
listaEquipos: Equipo[10]
totalEquipos: int
registrarEquipo()
registrarCorredor()
asignarPrincipal()
mostrarEquiposCorredores()
numeroExiste()
seleccionarEquipo()
seleccionarCorredor()
Carrera
nombre: String
vueltas: int
distancia: double
equipoGanador: String
corredorGanador: String
tiempoGanador: double
evalúa
evalúa
1
0..10
Equipo
id: String
nombre: String
performance: int
presupuesto: double
corredores: Corredor[2]
principal: Corredor
agregarCorredor()
tieneCorredores()
tieneMaxCorredores()
existeNumeroCorredor()
Data
pedirTexto()
pedirEntero()
pedirDecimal()
1
1
principal
0..2
0..1
Corredor
id: String
nombre: String
numero: int
habilidad: int
experiencia: int
equipoId: String
Métodos estáticos para validación de entrada

Avance 2

Diagrama actualizado



Avance 3

Documentación del Sistema F1 Manager, para el avance final.

1. Clase Equipo

Descripción:

Representa un equipo de Fórmula 1 con sus atributos y corredores asociados.

Atributos:

id (String): Identificador único (formato EQP-00001) - final

nombre (String): Nombre del equipo - final

performance (int): Nivel de performance (0-10) - final

presupuesto (double): Presupuesto en millones - final

corredores (Corredor[]): Array de 2 corredores máximo - final

principal (Corredor): Corredor principal del equipo

contador (static int): Contador para IDs autoincrementales

Métodos principales:

Constructor:

```
/**
 * Crea un nuevo equipo
 * @param nombre Nombre del equipo (no vacío)
 * @param performance Nivel (0-10)
```

* @param presupuesto En millones

*/

public Equipo(String nombre, int performance, double presupuesto)

Gestión de corredores:

public boolean agregarCorredor(Corredor corredor)

public boolean tieneCorredores()

public boolean tieneMaxCorredores()

public boolean existeNumeroCorredor(int numero)

public boolean contieneCorredorConNombre(String nombre)

Getters/Setters:

public String getId()

public String getNombre()

public int getPerformance()

public double getPresupuesto()

public Corredor[] getCorredores()

public Corredor getPrincipal()

public void setPrincipal(Corredor principal)

2. Clase Corredor

Descripción:

Representa un piloto de Fórmula 1 con sus atributos.

Atributos:

id (String): ID único (formato DR-00001) - final

nombre (String): Nombre completo - final

numero (int): Número del piloto - final

habilidad (int): Nivel (0-10) - final

experiencia (int): Nivel (0-10) - final

equipoId (String): ID del equipo asociado - final

contador (static int): Contador para IDs

Constructor:

public Corredor(String nombre, int numero, int habilidad,

int experiencia, String equipoId)

Getters:

public String getId()

public String getNombre()

public int getNumero()

public int getHabilidad()

public int getExperiencia()

public String getEquipoId()

3. Clase Data

Descripción:

Clase utilitaria para validación de datos.

Métodos:

public static String pedirTexto(String mensaje)

public static int pedirEntero(String mensaje, int min, int max)

public static double pedirDecimal(String mensaje)

4. Clase Equipos

Descripción:

Gestiona la colección de equipos y sus operaciones.

Atributos:

equipos (Equipo[]): Array de hasta 10 equipos

totalEquipos (int): Contador de equipos registrados

Métodos principales:

public void registrarEquipo()

public void registrarCorredor()

public void asignarPrincipal()

public void mostrarEquiposCorredores()

public Equipo[] getEquipos()

public int getTotalEquipos()

Métodos auxiliares:

private boolean existeNombreCorredorEnCualquierEquipo(String nombre)

private boolean numeroExiste(int numero)

private boolean existeNombreEquipo(String nombre)

private Equipo seleccionarEquipo(String mensaje, String titulo)

private Corredor seleccionarCorredor(Equipo equipo, String mensaje)

5. Clase F1_Manager

Descripción:

Clase principal con el método main().

Atributos:

gestorEquipos (static Equipos): Gestiona equipos

gestorCarreras (static Carreras): Gestiona carreras

gestorReportes (static GestorReportes): Genera reportes

Método main:

```
public static void main(String[] args) {
    // Menú interactivo con 7 opciones:
```

```
        // 1. Registrar equipo
        // 2. Registrar corredor
        // 3. Asignar principal
        // 4. Ver equipos/corredores
        // 5. Simular carrera
        // 6. Generar reportes
        // 7. Salir
    }
```

6. Clase GestorReportes

Descripción:

Genera reportes en formato CSV.

Métodos:

public void generarReportes(Equipos gestorEquipos, Carreras gestorCarreras)

private void generarReporteEquipos(Equipos gestorEquipos)

private void generarReporteCorredores(Equipos gestorEquipos)

private void generarReporteCarreras(Carreras gestorCarreras)

7. Diagrama de Clases (Resumen)

[Relaciones entre clases]:

- Equipo "contiene" 0-2 Corredores

- Equipos "gestiona" 0-10 Equipos

- F1_Manager "usa" Equipos, Carreras y GestorReportes

- GestorReportes "genera reportes de" Equipos y Carreras

8. Flujo Principal del Sistema

Inicio: Ejecución desde F1_Manager.main()

Menú principal: 7 opciones interactivas

Registros:

- Equipos (con validación de nombre único)
- Corredores (con validación de número y nombre único)

Operaciones:

- Asignación de corredor principal
- Visualización de datos
- Simulación de carreras

Reportes y terminar la ejecución:

- Generación automática de CSV’s
- Terminar programa

Diagrama final