Herencia, polimorfismo y colecciones

Realizado por Deiry Sofía Navas Muriel

Objetivos

LO QUE QUEREMOS LOGRAR

Herencia

La herencia permite que una clase “herede” los métodos y atributos de otra clase

Polimorfismo

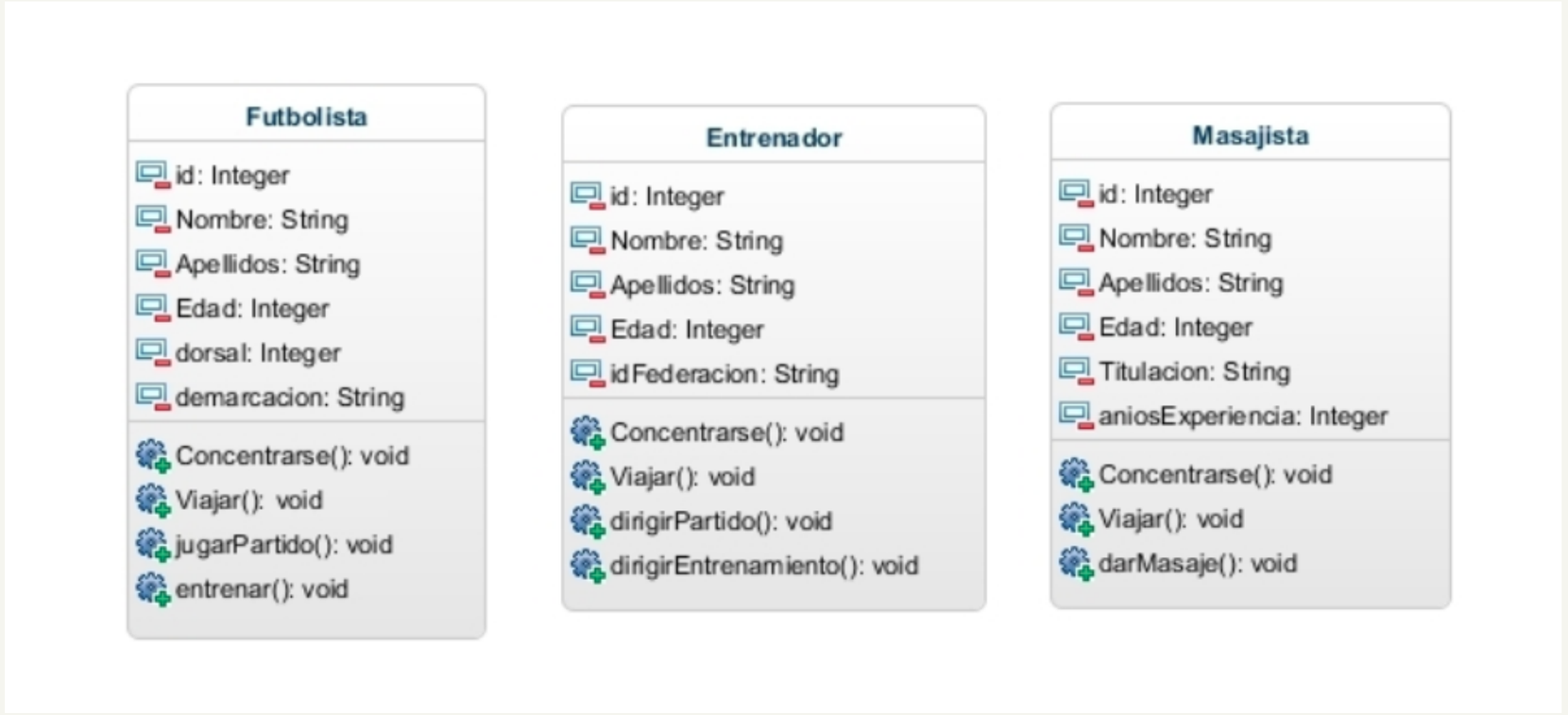
Permite llamar un método de un objeto dado, este funcione de manera diferente dependiendo de unas condiciones que veremos posteriormente.

Colecciones

Es un objeto que agrupa varios elementos del mismo tipo en una sola unidad. Las colecciones se suelen usar para almacenar, obtener, manipular y comunicar datos agrupados.

Herencia

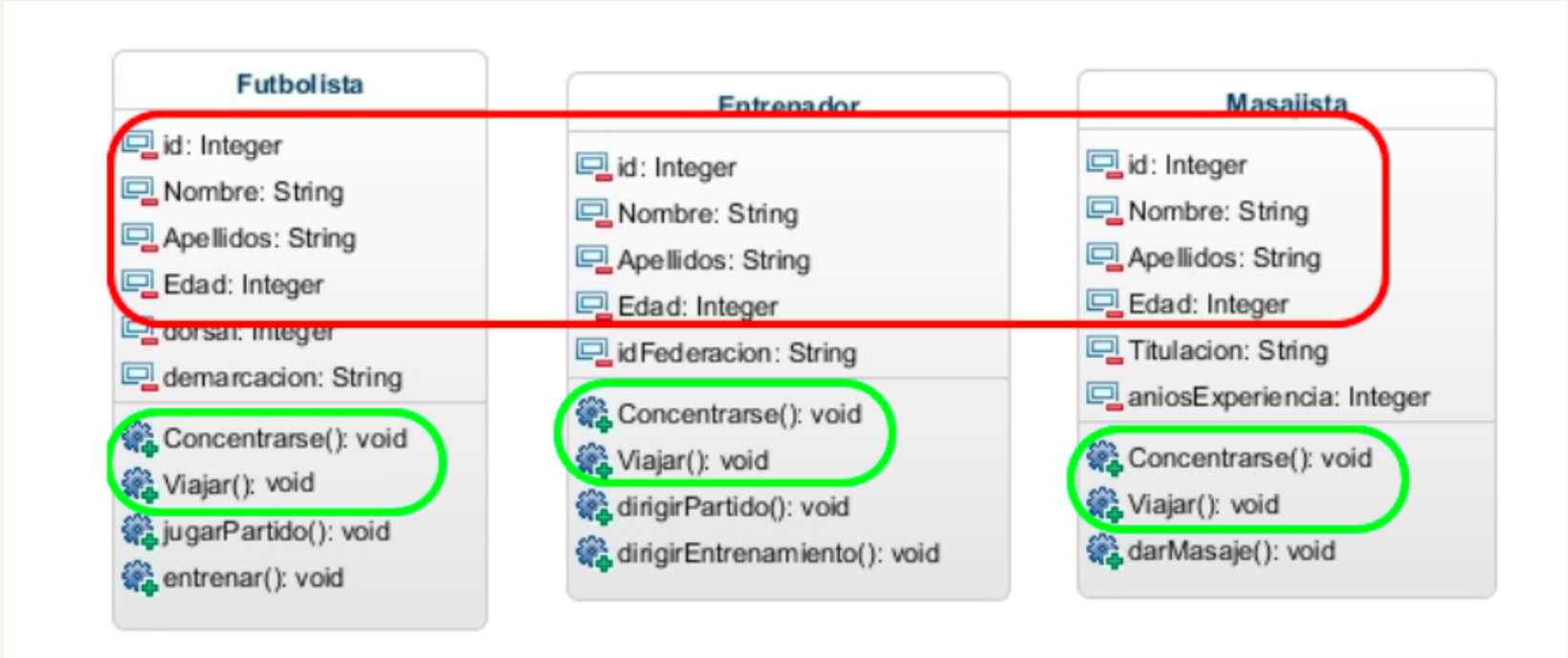
Vamos a hacer un ejercicio



Problema

LO QUE QUEREMOS RESOLVER

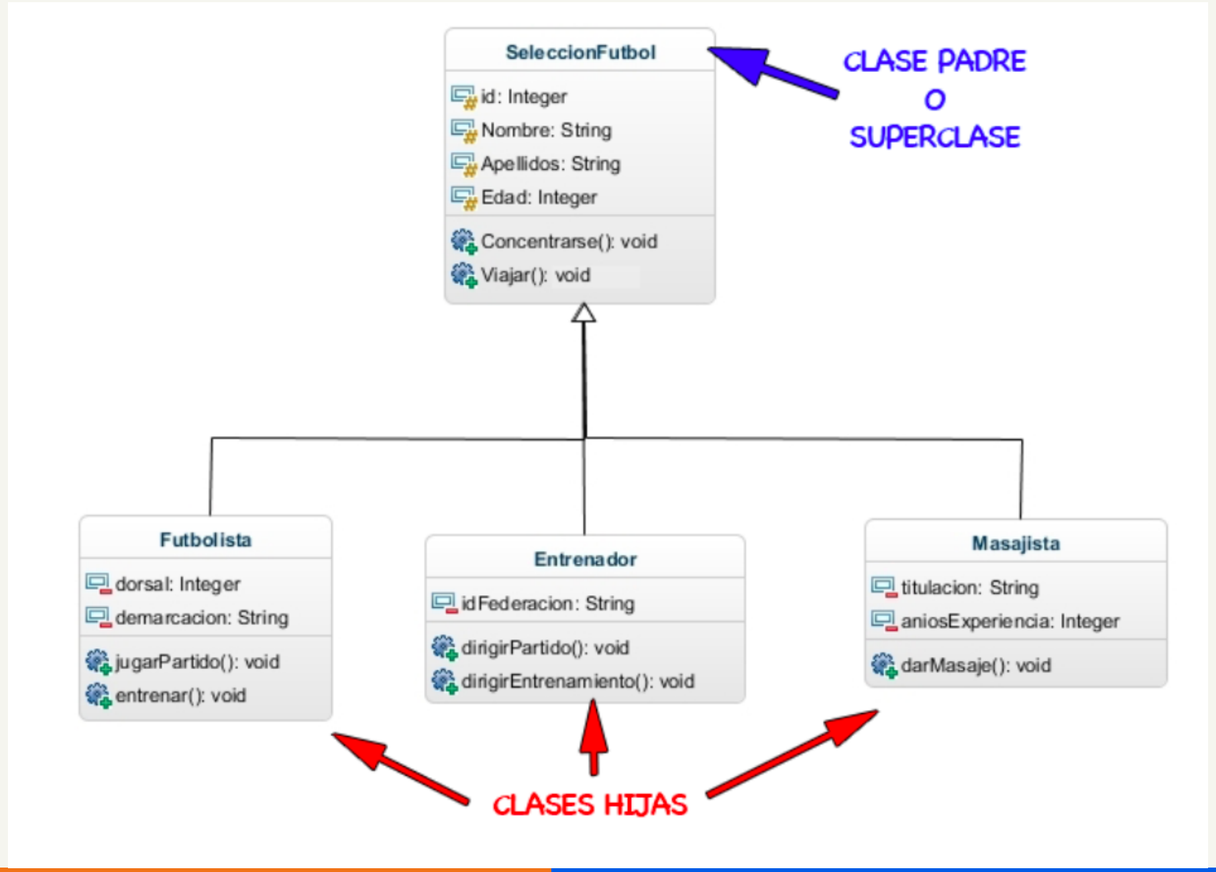
Como se puede observar, vemos que en las tres clases tenemos atributos y métodos que son iguales ya que los tres tienen los atributos id, Nombre, Apellidos y Edad ; y los tres tienen los métodos de Viajar y Concentrarse:



Problema

LO QUE QUEREMOS RESOLVER

Crear una clase con el "código que es común a las tres clases" (a esta clase se le denomina en la herencia como "Clase Padre o SuperClase") y el código que es especifico de cada clase , lo dejaremos en ella, siendo denominadas estas clases como "Clases Hijas", las cuales heredan de la clase padre todos los atributos y métodos públicos o protegidos . Es muy importante decir que las clases hijas no van a heredar nunca los atributos y métodos privados de la clase padre , así que mucho cuidado con esto.



Términos para tener en cuenta

EXTENDS

Esta palabra reservada, indica a la clase hija cual va a ser su clase padre, es decir que por ejemplo en la clase Futbolista al poner "public class Futbolista extends SeleccionFutbol" le estamos indicando a la clase 'Futbolista' que su clase padre es la clase 'SeleccionFutbol'

PROTECTED

sirve para indicar un tipo de visibilidad de los atributos y métodos de la clase padre y significa que cuando un atributo es 'protected' o protegido, solo es visible ese atributo o método desde una de las clases hijas y no desde otra clase.

SUPER

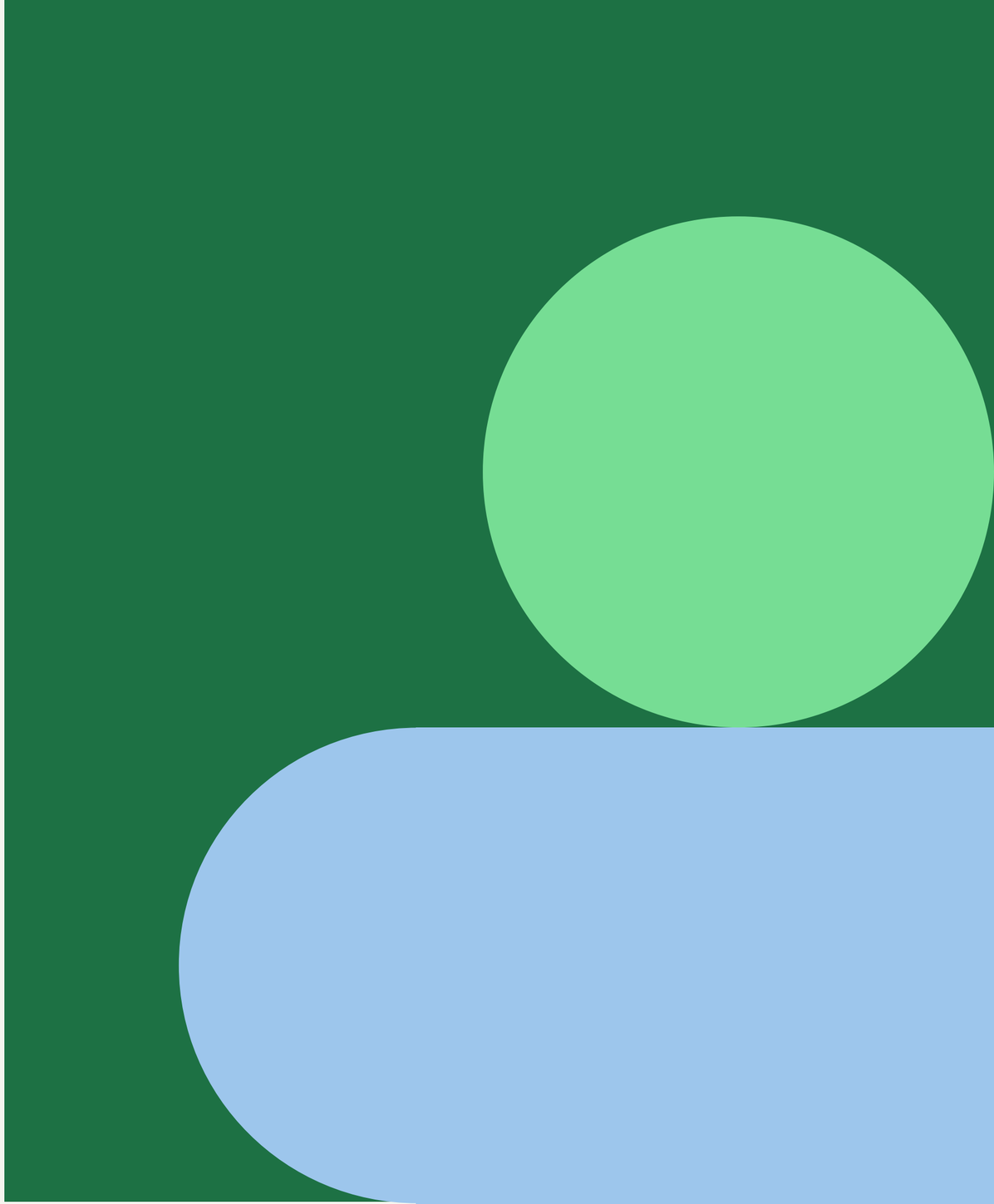
sirve para llamar al constructor de la clase padre.

```
public class Futbolista extends SeleccionFutbol {
        ......
        public Futbolista() {
                super();
        }

        public Futbolista(int id, String nombre, String a
                super(id, nombre, apellidos, edad);
                this.dorsal = dorsal;
                this.demarcacion = demarcacion;
        }
```

Polimorfirmo

LOS CONCEPTOS DE JAVA POLIMORFISMO Y HERENCIA ESTÁN INTIMAMENTE RELACIONADOS



MISION TIC

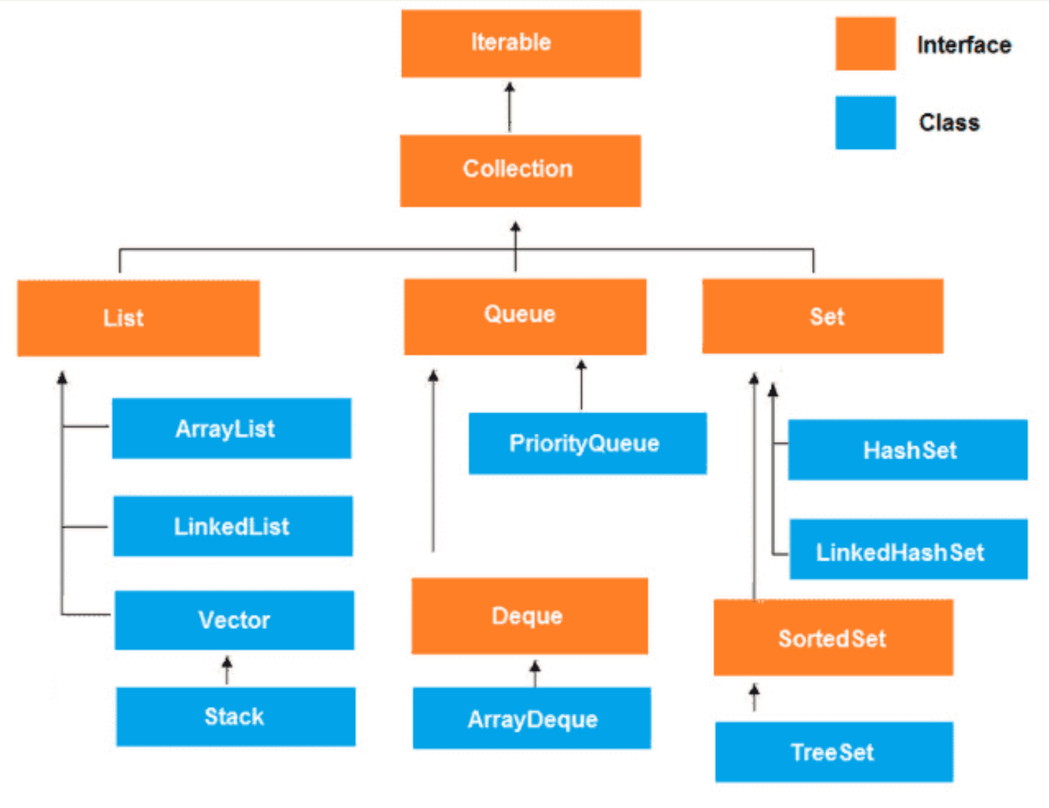
El concepto de polimorfismo

01 Nos ayuda a la hora de generar flexibilidad en el código pero sobre todo también a la hora de simplificar el número de conceptos que un programador debe manejar.

02 Es la capacidad de un objeto de adquirir varias formas. El uso más común de polimorfismo en programación orientada a objetos se da cuando se utiliza la referencia de una clase padre, para referirse al objeto de la clase hijo.

03 Es una característica de la programación orientada a objetos que permite llamar a métodos con igual nombre pero que pertenecen a clases distintas.

Colecciones



wrapper class

01 Una clase de envoltura o wrapper class es una clase que se puede usar en lugar de un tipo de datos primitivo y cuyos objetos (variables) funcionan de manera similar a las variables normales, más específicamente, de manera similar a las de tipo String. Cada tipo de datos primitivo tiene una wrapper class equivalente y en todos los casos el valor por defecto será null, tal como se observa a continuación.

Tipo primitivo	Wrapper class
boolean	Boolean
byte	Byte
short	Short
char	Character
int	Integer
long	Long
float	Float
double	Double

Herencia, polimorfismo y colecciones

Deiry Sofía Navas Muriel

Estudiante Ing. de sistemas

Herencia, polimorfismo y colecciones

MISION TIC