Un enumerado (o Enum) es una clase "especial" (tanto en Java como en otros lenguajes) ***que limitan la creación de objetos a los especificados explícitamente en la implementación de la clase.*** La única limitación que tienen los enumerados respecto a una clase normal es que si tiene constructor, este debe de ser privado para que no se puedan crear nuevos objetos.

Vamos a empezar con un sencillo ejemplo e una clase Enum. Volviendo a los ejemplo relacionados con el fútbol, tenemos que lo futbolistas están caracterizados por una demarcación a la hora de jugar un partido de fútbol, por tanto las demarcaciones en las que puede jugar un futbolista son finitas y por tanto se pueden enumerar en: Portero, Defensa, Centrocampista y Delantero. Con esta especificación podemos crearnos la siguiente clase "Enum" llamada "Demarcación":

```
public enum Demarcacion
{
    PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO
}
```

Por convenio (aunque lo podéis poner como queráis) los nombres de los enumerados se escriben en mayúsculas.

Es muy importante entender que un "***Enum***" en java ***es realmente una clase*** (cuyos objetos solo pueden ser los definidos en esta clase: PORTERO, …, DELANTERO) ***que hereda*** de la clase "***Enum(java.lang.Enum)***" y por tanto los enumerados tienen una serie de métodos heredados de esa clase padre (Pulsar AQUI para ver su JavaDoc). A continuación vamos a mostrar algunos de los métodos más utilizados de los enumerados:

```
public enum Demarcacion{PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO}
Demarcacion delantero = Demarcacion.DELANTERO;    // Instancia de un enum de la clase Demarcación
delantero.name();    // Devuelve un String con el nombre de la constante (DELANTERO)
delantero.toString();    // Devuelve un String con el nombre de la constante (DELANTERO)
delantero.ordinal();    // Devuelve un entero con la posición del enum según está declarada (3).
delantero.compareTo(Enum otro);    // Compara el enum con el parámetro según el orden en el que están declarados lo enum
Demarcacion.values();    // Devuelve un array que contiene todos los enum
```

Visto cuales son los métodos más utilizados dentro de los enumerados, vamos a poner un ejemplo para ver los resultados que nos devuelven estos métodos. Dado el siguiente fragmento de código:

```
Demarcacion delantero = Demarcacion.DELANTERO;
Demarcacion defensa = Demarcacion.DEFENSA;

// Devuelve un String con el nombre de la constante
System.out.println("delantero.name()= "+delantero.name());
System.out.println("defensa.toString()= "+defensa.toString());

//  Devuelve un entero con la posición de la constante según está declarada.
System.out.println("delantero.ordinal()= "+delantero.ordinal());

// Compara el enum con el parámetro según el orden en el que están declaradas las constantes.
System.out.println("delantero.compareTo(portero)= "+delantero.compareTo(defensa));
System.out.println("delantero.compareTo(delantero)= "+delantero.compareTo(delantero));

// Recorre todas las constantes de la enumeración
for(Demarcacion d: Demarcacion.values()){
        System.out.println(d.toString()+" - ");
}
```

Tenemos como salida los siguientes resultados:

```
delantero.name()= DELANTERO
defensa.toString()= DEFENSA
delantero.ordinal()= 3
delantero.compareTo(defensa)= 2
delantero.compareTo(delantero)= 0
PORTERO - DEFENSA - CENTROCAMPISTA - DELANTERO
```

Como ya se ha dicho un enum es una clase especial que limita la creación de objetos a los especificados en su clase (por eso su constructor es privado, como se ve en el siguiente fragmento de código); pero estos objetos pueden tener atributos como cualquier otra clase. En la siguiente declaración de la clase, vemos un ejemplo en la que definimos un enumerado "Equipo" que va a tener dos atributos; el nombre y el puesto en el que quedaron en la liga del año 2009/2010.

```
public enum Equipo
{
	BARÇA("FC Barcelona",1), REAL_MADRID("Real Madrid",2),
	SEVILLA("Sevilla FC",4), VILLAREAL("Villareal",7);

	private String nombreClub;
	private int puestoLiga;

	private Equipo (String nombreClub, int puestoLiga){
		this.nombreClub = nombreClub;
		this.puestoLiga = puestoLiga;
	}

	public String getNombreClub() {
		return nombreClub;
	}

	public int getPuestoLiga() {
		return puestoLiga;
	}

}
```

Como se ve BARÇA, REAL_MADRID, etc. son el nombre del enumerado (u objetos de la clase Equipo) que tendrán como atributos el "nombreClub" y "puestoLiga". Como se ve en la clase ***definimos un constructor que es privado*** (es decir que solo es visible dentro de la clase Equipo) y solo definimos los métodos "get". Para trabajar con los atributos de estos enumerados se hace de la misma manera que con cualquier otro objeto; se instancia un objeto y se accede a los atributos con los métodos get. En el siguiente fragmento de código vamos a ver como trabajar con enumerados que tienen atributos:

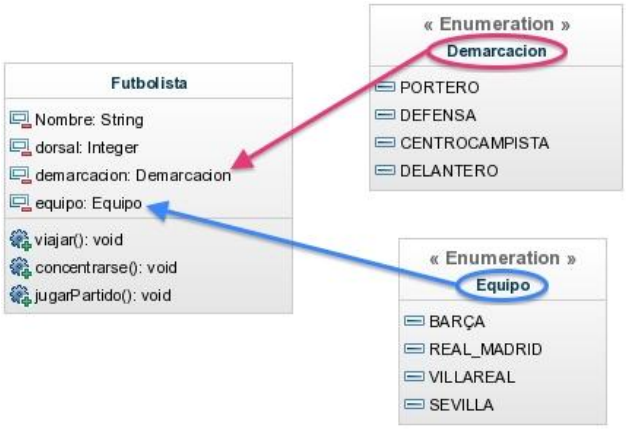
```
// Instanciamos el enumerado
Equipo villareal = Equipo.VILLAREAL;

// Devuelve un String con el nombre de la constante
System.out.println("villareal.name()= "+villareal.name());

// Devuelve el contenido de los atributos
System.out.println("villareal.getNombreClub()= "+villareal.getNombreClub());
System.out.println("villareal.getPuestoLiga()= "+villareal.getPuestoLiga());
```

Como salida de este fragmento de código tenemos lo siguiente:

```
villareal.name()= VILLAREAL
villareal.getNombreClub()= Villareal
villareal.getPuestoLiga()= 7
```

Es muy importante que tengáis claro que los enumerado no son Strings (aunque pueden serlo), sino que son objeto de una clase que solo son instanciables desde la clase que se implementa y que no se puede crear un objeto de esa clase desde cualquier otro lado que no sea dentro de esa clase. Es muy común (sobre todo cuando se esta aprendiendo que son los enumerados) que se interprete que un enumerado es una lista finita de Strings y en realidad es una lista finita de objetos de una determinada clase con sus atributos, constructor y métodos getter aunque estos sean privados.

A continuación vamos a poner un sencillo ejemplo en el que vamos a mezclar los dos enumerados anteriores (Demarcación y Equipo). En este ejemplo (siguiendo la línea de ejemplo puestos en las entradas de Herencia, Polimorfismo y Polimorfismo II) vamos a crearnos unos objetos de la clase Futbolista, que representarán a los jugadores de la selección española de fútbol que ganaron el mundial de fútbol de Sudáfrica en el año 2010. Esta clase va a caracterizar a los futbolistas por su nombre, su dorsal, la demarcación en la que juegan y el club de fútbol al que pertenecen; tal y como vemos en el siguiente diagrama de clases.



Como vemos los atributos de demarcación y equipo son de la clase Demarcacion y Equipo respectivamente y son los enumerados vistos anteriormente; por tanto un futbolista solo podrá pertenecer a uno de los cuatro equipos que forman el enumerado "Equipo" y podrá jugar en alguna de las cuatro demarcaciones que forman el enumerado "Demarcación". A continuación mostramos la implementación de la clase "Futbolista":

```
package Main;

public class Futbolista {

	private int dorsal;
	private String Nombre;
	private Demarcacion demarcacion;
	private Equipo equipo;

	public Futbolista() {
	}

	public Futbolista(String nombre, int dorsal, Demarcacion demarcacion, Equipo equipo) {
		this.dorsal = dorsal;
		Nombre = nombre;
		this.demarcacion = demarcacion;
		this.equipo = equipo;
	}

	// Metodos getter y setter

	.............................


	@Override
	public String toString() {
		return this.dorsal + " - " + this.Nombre + " - "
				+ this.demarcacion.name() + " - " + this.equipo.getNombreClub();
	}
}
```

Dada esta clase podemos crearnos ya objetos de la clase futbolista, como mostramos a continuación:

```
Futbolista casillas = new Futbolista("Casillas", 1, Demarcacion.PORTERO, Equipo.REAL_MADRID);
Futbolista capdevila = new Futbolista("Capdevila", 11, Demarcacion.DEFENSA, Equipo.VILLAREAL);
Futbolista iniesta = new Futbolista("Iniesta", 6, Demarcacion.CENTROCAMPISTA, Equipo.BARÇA);
Futbolista navas = new Futbolista("Navas", 22, Demarcacion.DELANTERO, Equipo.SEVILLA);
```

Como vemos la demarcación y el equipo al que pertenecen solo pueden ser los declarados en la clase enumerado. Si llamamos al método "toString()" declarado en la clase futbolista, podemos imprimir por pantalla los datos de los futbolistas. Dado el siguiente código:

```
System.out.println(casillas.toString());
System.out.println(capdevila.toString());
System.out.println(iniesta.toString());
System.out.println(navas.toString());
```

Y dado el siguiente método "toString()"

```
@Override
public String toString() {
        return this.dorsal + " - " + this.Nombre + " - "
                        + this.demarcacion.name() + " - " + this.equipo.getNombreClub();
}
```

Tenemos como salida lo siguiente:

```
1 - Casillas - PORTERO - Real Madrid
11 - Capdevila - DEFENSA - Villareal
6 - Iniesta - CENTROCAMPISTA - FC Barcelona
22 - Navas - DELANTERO - Sevilla FC
```

En resumen esto es todo lo importante que debes saber sobre los enumerados en Java. En otros lenguajes de programación los enumerados pueden tener más métodos heredados, pero el concepto fundamental de un enumera es que son unos objetos (y no Strings; aunque es si en java un String es un clase y no un dato atómico) definidos en la misma clase con constructor privado y si tiene atributos estos solo tienen que tener métodos "getter" para obtener el valor del atributo.