INTERFACES

Una interfaz es un conjunto de métodos abstractos y propiedades, en ella se especifica que se debe hacer pero no se permite su implementación.

Modo de Uso:

Java nos proporciona dos palabras reservadas para trabajar con interfaces:

- **Interface**
- **Implements**

Forma de declarar una interfaz:

modificador_acceso **interface** NombreInterfaz {

....

}

modificador_acceso puede ser una clase de objetos que permite utilizar herencia en abstracción constante a las clases en las que se implementen.

Forma de implementar una clase:

modificador_acceso **class** NombreClase **implements** NombreInterfaz1 [, NombreInterfaz2]

- Una clase puede implementar varias interfaces solo separando los nombres con coma.

Ejemplo:

Definición de una interfaz:

```
interface Nave
{
    public static final int VIDA = 100;

    public abstract void moverPosicion (int x, int y);
    public abstract void disparar();
    .....
}
```

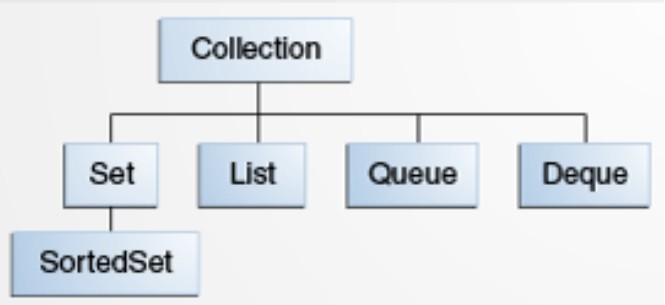
Uso de la interfaz definida:

```
public class NaveJugador implements Nave
{
```

```
    public void moverPosicion (int x, int y)
    {
        //Implementación del método
    }
    public void disparar()
    {
        //Implementación del método
    }
    .....
}
```

En forma común, una interfaz es un grupo de métodos relacionados con cuerpos vacíos.

Interfaz Collection



El interfaz collection permite básicamente añadir, eliminar y recorrer la estructura gracias a un iterator.

El interfaz List mejora el comportamiento de collection permitiendo extraer, buscar y reemplazar ocurrencias.

El List posee un orden de almacenamiento asegurando así que los elementos se mantengan en orden determinado.

El List añade un conjunto de métodos a collection que permiten insertar y borrar elementos en mitad de lista.

El interfaz Set no amplia el comportamiento de collection.

El interfaz Set es único para cada elemento añadido.

Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

Elementos de un patrón:

- Nombre: describe el problema de diseño.
- El problema: describe cuándo aplicar el patrón.
- La solución: describe los elementos que componen el diseño, sus relaciones, responsabilidades y colaboración.

Clases de patrones:

- De creación: conciernen al proceso de creación de objetos.
- De estructura: tratan la composición de clases y/o objetos.
- De comportamiento: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

Abstract Factory

El patrón Abstract Factory nos permite crear, mediante una interfaz, conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cuál es el objeto concreto.

Este patrón se puede aplicar cuando:

- Un sistema debe ser independiente de cómo sus objetos son creados.
- Un sistema debe ser 'configurado' con una cierta familia de productos.
- Se necesita reforzar la noción de dependencia mutua entre ciertos objetos.

Elementos del Patrón Abstract Factory

FabricaAbstracta*****: Define un conjunto de métodos (interfaz) para la creación de productos abstractos.

- FabricaConcreta1/2**:** Implementa la interfaz de la FabricaAbstracta para la creación de los distintos productos concretos.
- ProductoAbstractoA*/B***:** Define la interfaz de los objetos de tipo ProductoA/B.
- ProductoConcretoA1/A2/B1/B2: Implementan su respectiva interfaz representando un producto concreto.

Estructura del patrón Abstract Factory