# Colecciones e interfaces

## 1 COLECCIONES

Las colecciones son conjuntos (objetos)  de objetos; se diferencian de los arreglos en que estos no tienen un tamaño fijo lo cuál los vuelve dinámicos.

Las colecciones las encontramos en el paquete java.util

### • INTERFAZ (- - - → ) <<INTERFACE>>

Una interfaz es un tipo, un modelo de una clase se utiliza para alcanzar la plena abstracción, es un medio de comunicación , Las interfaces son usadas para la implementación.

Una interfas separa las especificaciones de una clase (qué hace) de la implementación (cómo se hace), esto hace que se generen programas con menos errores.

Las interfaces más conocidas son :

SET Una colección donde no puede haber elementos repetidos, cuyos elementos se almacenan no necesariamente siguiendo un orden particular.

LIST Una colección que posee un orden particular a menos que se modifique la lista, puede contener elementos duplicados

MAP

UN objeto que asocia claves con valores.

No puede tener claves duplicadas

Una interface es una variante de una clase abstracta con la condición de que todos sus métodos deben ser asbtractos. Si la interface va a tener atributos, éstos deben llevar las palabras reservadas static final y con un valor inicial ya que funcionan como constantes por lo que, por convención, su nombre va en mayúsculas.

interface Nomina

{

```
 public static final String EMPRESA = "Patito, S. A.";

 public void detalleDeEmpleado(Nomina obj);

}
```

- **CLASES ABSTRACTAS**

  Una clase que declara la existencia de métodos pero no la implementación de dichos métodos (o sea, las llaves { } y las sentencias entre ellas), se considera una clase abstracta.
  Una clase abstracta puede contener métodos no-abstractos pero al menos uno de los métodos debe ser declarado abstracto.

  Para declarar una clase o un metodo como abstractos, se utiliza la palabra reservada **abstract**.

```
abstract class Drawing
{
  abstract void miMetodo(int var1, int var2);
  String miOtroMetodo( ){ ... }
}
```

  Una clase abstracta no se puede instanciar pero si se puede heredar y las clases hijas serán las encargadas de agregar la funcionalidad a los métodos abstractos. Si no lo hacen así, las clases hijas deben ser también abstractas.