#### **Package**

En una clase, package hace referencia al acceso via path en referencia al proyecto.

package agenda.modelo;

En este ejemplo indica que la carpeta agenda(en **src**(source)del proyecto hay una subcarpeta que se llama modelo y todas las clases que estén en la carpeta modelo podrán acceder a esta clase(cuando se importan).

#### **Imports**

Detalla la ubicación de **clases externas en uso** en la clase, indicando también el path.

**Java.util** es la librería por defecto de Java con la principales clases(String,Integer…)

import java.util.LinkedHashSet;

import java.util.Objects;

import java.util.Set;

import java.util.TreeSet;

#### **Declaración de la clase**

Contiene en un orden concreto:

1. **modificador de acceso:** (public, protected, private, default) En una clase siempre será public ya que si no no podrán usarse.
2. **nombre de la clase:** ( siempre en CamelCase)
3. opcionalmente, **palabras clave que indican propiedades sobre la clase**: (implement, extends, etc)

public class Contacto implements Comparable<Contacto> {

#### **Cuerpo de la clase**

##### Los atributos:

Son las propiedades de una clase y serán siempre private, lo que encapsula para evitar modificación externa.

//Atributos

private int idContacto;

private String nombre;

private String apellidos;

private String apodo;

private Domicilio dom;

private Set<String> telefonos;

private Set<String> correos;

##### Los constructores:

Son los que indican como serán los objetos construidos de la clase, tambien serán públicos. Puede haber mas de un constructor pero el principal es el que está primero.

//Constructor

public Contacto() {

telefonos = new LinkedHashSet<String>();//por orden de carga, por eso LinkedHashSet

correos = new LinkedHashSet<String>();//por orden de carga, por eso LinkedHashSet

}

public Contacto(String nombre, String apellidos, String apodo, Domicilio dom, Set<String> telefonos,

Set<String> correos, int idContacto) {

this();//hace una llamada al constructor por defecto

this.nombre = nombre;

this.apellidos = apellidos;

this.apodo = apodo;

this.dom = dom;

this.idContacto = idContacto;

// Lee estos atributos del constructor principal

// this.telefonos = telefonos;

// this.correos = correos;

}

##### Los getters y setters:

Hacen referencia a los atributos de la clase.

Getter devuelve los valores default de las propiedades de la clase.

Setter generalmente recibe parametro y modificará los atributos de la clase y mediante this.parametroDeLaClase dará valor.

//Getters y Setters

public int getIdContacto() {

return idContacto;

}

public void setIdContacto(int idContacto) {

this.idContacto = idContacto;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getApellidos() {

return apellidos;

}

public void setApellidos(String apellidos) {

this.apellidos = apellidos;

}

public String getApodo() {

return apodo;

}

public void setApodo(String apodo) {

this.apodo = apodo;

}

public Domicilio getDom() {

return dom;

}

public void setDom(Domicilio dom) {

this.dom = dom;

}

public Set<String> getTelefonos() {

return telefonos;

}

// public void setTelefonos(Set<String> telefonos) {

// this.telefonos = telefonos;

// }

//Este metodo sustituiría a set telefonos permitiendo la insercion de varios telefonos

public void addTelefonos(String... telefonos) {

for (String telefono : telefonos) {

this.telefonos.add(telefono);

}

}

public Set<String> getCorreos() {

return correos;

}

// public void setCorreos(Set<String> correos) {

// this.correos = correos;

// }

//Este metodo sustituiría a set correos permitiendo la insercion de varios correos

public void addCorreos(String... coreeos) {

for (String correo : correos) {

this.correos.add(correo);

}

}

#### **Overrides**

Hacen referencia a los métodos que heredan y se quieren sobreescribir para modificar el comportamiento, generalmente suelen ser: equals para comparar objetos y no referenciar lugares en memoria, hashCode que utiliza el hash del objeto(id del objeto que se genera al crear dicho objeto),toString para elegir que información queremos mostrar del objeto.

//Sobreescritura hashCode, equals y toString

@Override

public int hashCode() {

// return Objects.hash(idContacto);

return idContacto;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Contacto other = (Contacto) obj;

return idContacto == other.idContacto;

}

@Override

public String toString() {

return "Contacto [idContacto=" + idContacto + ", nombre=" + nombre + ", apellidos=" + apellidos + ", apodo="

+ apodo + ", dom=" + dom + ", telefonos=" + telefonos + ", correos=" + correos + "]";

}

//para definir el orden natural de contactos

@Override

public int compareTo(Contacto o) {

return this.idContacto - o.idContacto;

}

#### **Métodos propios de la clase**

Serán los métodos que se definen en la clase, generalmente privados si son para funcionamiento interno de la clase.

}