Maven es el que se encarga de cargar las bibliotecas. Java es un lenguaje compilado para poder ejecutarlo lo tengo que compilar

Casteo

No es que le cambia el tipo a la variable sino que lo considera como que fuera otra cosa.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Herencia

La herencia es rígida, tengo una herencia única.

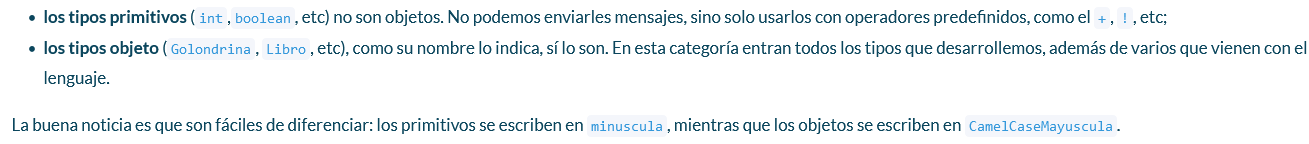


**ATRIBUTOS**

¿Como usar los atributos? Si yo quiero usar un atributo hago por ejemplo this.atributo , sin los paréntesis. Los paréntesis se usan en java solamente en los métodos.

Valores por defecto Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Tipos Objetos

This = self

Modificadores de acceso

Tabla

Descripción generada automáticamente

Las clases las vamos a escribir SIEMPRE como PUBLICAS.

Los atributos SIEMPRE van a ser PRIVADOS. (a veces pueden ser protegidos)

Poner algo como privado, solo puede acceder a eso la misma clase. Si yo lo tengo como protegido también pueden acceder las clases hijas. Esa es la dif entre privado y protegido.

En todos los casos tenemos que anotar los modificadores de acceso, salvo en las interfaces que no hace falta pq si no lo anotamos por defecto es public.

¿Como consigo un objeto?

con la función new(). No es necesario siempre darle nombre.

Variables de clase



Es una variable que la tengo a nivel clase no a nivel instancia, si se llega a modificar ese valor afecta a TODAS las instancias. Ej yo tengo muchos autos instanciados con una variable

Private static int anioAuto;

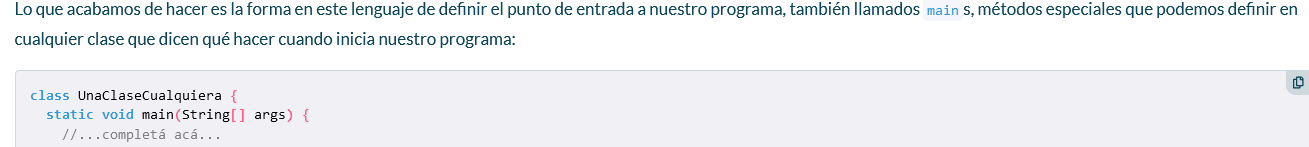
Tengo n autos 2011 si alguno de esos autos cambia el valor todas las instancias me van a cambiar de valor

Así como existen variables de clase, existen variables de método

Super

Texto

Descripción generada automáticamente

Definir el punto de entrada a nuestro programa

Texto

Descripción generada automáticamente

Constructores

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Los constructores siempre se llaman igual que la clase y nunca tienen retorno.

Ej de uso

Libro harryPotter = new Libro(2000,false);

INTERFAZ

Texto

Descripción generada automáticamente

Cuando declaramos una interfaz tenemos que listar todos los métodos que la componen, pero no sus implementaciones.

Las interfaces también nos permiten crear nuevos tipos.

Se puede hacer que una interfaz heredede de otra.

Los métodos no llevan cuerpo solamente firma. Si no hay comportamiento repetido hago una interfaz en lugar de una clase abstracta

COLECCIONES

List

es una interfaz que extiende de la interfaz collection. Tiene orden, tiene repetidos.

Set

No tiene repetidos, no tiene orden

List, Set y Map son interfaces, estas son sus clases.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente con confianza media

Como nuestras colecciones son objetos tenemos que inicializarlas

Stream() es un objeto que se agrego a las colecciones que viene a agregar unos métodos de colección. Cuando nosotros llamamos a stream y a algún método de strem ej(filter,map,flatmap) esos métodos van a devolver nuevamente un stream, si lo queremos convertir en una colección tenemos que llamar al método collect.

Hay varios tipos de stream, stream es la superclase, pero existe también IntStream.

FILTER no devuelve una colección, si solo los quiero contar puedo hacer un count en cambio de un size.

CLASES ABSTRACTAS

No pueden ser instanciadas. Son como las interfaces se usan en lugar de las interfaces cuando hay lógica común.

Como no hay lógica común entre los distintos estados, *no hay necesidad de definir una superclase* (abstracta), sino que todos ellos **comparten la misma interfaz**, lo que quiere decir que entienden el mismo conjunto de mensajes.

Se usa una clase abstracta en cambio de una interfaz si hay lógica compartida. No es argumento si hay muchos atributos compartidos, eso no tiene nada que ver.

Si no hay lógica compartido tenes que hacer una interfaz

Diagrama

Descripción generada automáticamente

Imagen que contiene texto, dibujo

Descripción generada automáticamente

Los repositorios son abstracciones de almacenamiento

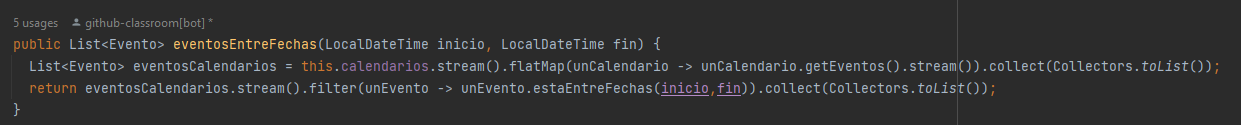
Pueden tener lógica pero solo de filtrado y devolver cosas esa onda. No puede tener lógica de conectarse a una API por ejemplo.

Si una clase lo único que hace es ejecutar cosas pero no guarda estados ni lista de estados me conviene que sea singleton.

Los singleton lo bueno es que no necesito que lo conozca cada clase como atributo ya lo conoce el sistema y lo puedo usar directamente.

Problema con el map Hay que agregarle el stream adentro de la función lamba sino tira error.

Te tira este error si no lo agregas “no instance(s) of type variable(s) R exist so that List<Evento> conforms to Stream<? extends R>”



**MOCK**

lo que nosotros queremos probar no tiene que estar mockeado ej clase mailSender, las cosas que usa y no queremos probar (ej sendGrind) de esa clase es lo que tenemos que mockear.

Se lo mockeamos en el constructor.

Ej de mockeo

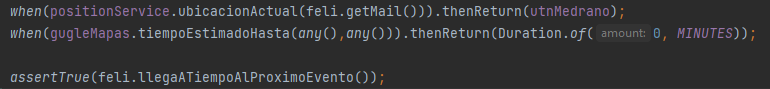
Primero lo que hay que hacer es

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Hay 2 metodos para mockear: When y verify.

El when sirve para cuando llamo a un método que me haga tal cosa. EJ



El verify sirve para ver si ese método fue llamado alguna vez

Verify(elMock,atMostOnce()).elMetodoQueQuieroVerSiSellamo)

Texto

Descripción generada automáticamente

**¿Cómo testear una Excepción?**

AssertThrows(NombreExcepcion.class , () -> metodoQueMeTiraEsaExcepcion)

Texto

Descripción generada automáticamente

**¿Como buscar el mínimo de una Colección?**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**COMO HACER SORT**

