

CÓMO SUBIR PROYECTO A GIT (OJO!!! LO EXPLICAN UNA SOLA VEZ PERO LO VAN A TOMAR):

1- CREAR EL REPO EN GIT (¡publico!)

2- CREAR EL PROYECTO EN ECLIPSE

3- ABRIR GIT CMD Y HACER LO SIGUIENTE:

C:

cd (pegar la ruta del proyecto de eclipse)

Cómo sacar la ruta: click derecho en el proyecto -> propiedades -> copiar donde dice Location

git init

git status

4- SINCRONIZAR EL PROYECTO CON EL REPO:

git remote add origin https://github.com/tucuenta/nombredelrepo.git

git branch -M main

git add .

git commit -m "primer commit"

git push -u origin main

5- ACTUALIZAR CAMBIOS

git add .

git commit -m "explicar qué cambio se hizo"

git push -u origin main

COSAS DE BÁSICA 1:**INGRESAR UNA FECHA CON EL TIPO LOCALDATE**

LocalDate.of(2008, 11, 24);

REDONDEAR NUMERO A X CANTIDAD DE DECIMALES

String.format("%.1f", numero) (1 decimal)

RECORTAR STRING (lo van a tomar para generar IDs en base a un string):

cadena.substring(posición en la que empieza) RECORDAR QUE ARRANCA EN CERO

CONVERTIR DE STRING A INTEGER:

Integer.valueOf(cadena)

TRANSFORMAR UNA LISTA DESORDENADA EN UN SET ORDENADO (LIST → TREESSET)

Empleados es una Lista. Creamos un nuevo TreeSet y copiamos todo el contenido de la Lista al TreeSet para que se ordenen automáticamente (HAY QUE IMPLEMENTAR COMPARABLE PARA DECIR EL CRITERIO)

```
Set<Empleado> setOrdenado = new TreeSet<>();  
setOrdenado.addAll(listaDesordenada);
```

ojo! Al pasar a un SET no permite hacer get(índice) porque no es de acceso secuencial

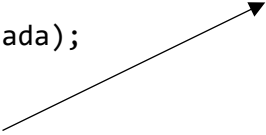
IMPLEMENTAR COMPARABLE CUANDO SE USA TREE-

```
public class Empleado implements Comparable<Empleado>  
{  
    @Override  
    public int compareTo(Empleado o) {  
        return this.id.compareTo(o.id);  
    }  
}
```

OJO! Si ya tenía un criterio de ordenamiento, no puedo hacer Override del compareTo otra vez, entonces creo una clase que implemente Comparator y la paso como parámetro al TreeSet:

IMPLEMENTAR COMPARATOR CUANDO SE USA TREE-

```
Set<Empleado> setOrdenado = new TreeSet<>(new OrdenEmpleadoPorNombre());  
setOrdenado.addAll(listaDesordenada);
```



```
public class OrdenEmpleadoPorNombre implements Comparator<Empleado>{  
    @Override  
    public int compare(Empleado o1, Empleado o2) {  
        if (o1.getNombre().compareTo(o2.getNombre()) == 0) {  
            // si los nombres son iguales, dame el del ID mayor primero  
            return o2.getId().compareTo(o1.getId());  
        } else {  
            // dame el del nombre menor  
            return o1.getNombre().compareTo(o2.getNombre());  
        }  
    }  
}
```

ORDENAR UNA LISTA DE OBJETOS EN BASE A OTRO OBJETO CON UN MAP

(por cada generación distinta quiero una lista de empleados que pertenecen a esa generación)

```
Map<Generacion, List<Empleado>> empleadosPorGeneracion = new TreeMap<>();

for (Generacion generacion : generaciones) {
    List <Empleado> listaEmpleados= obtenerListaEmpleadosDeUnaGeneracion(generacion);
    empleadosPorGeneracion.put(generacion, listaEmpleados);
}
```

ACCEDER A UN VALOR DEL MAP MEDIANTE LA CLAVE:

```
mapa.get(clave);
```

MOSTRAR UN MAPA:

```
syso(mapa.entrySet());
```

MOSTRAR UN MAPA QUE TIENE UNA LISTA COMO VALOR CON UN CICLO FORMAP:

```
for (Entry<Generacion, List<Empleado>> entry : mapa.entrySet()) {
    Generacion key = entry.getKey();
    List<Empleado> val = entry.getValue();
    for (Empleado empleado : val) {
        System.out.println(key + ": " + empleado.toString());
    }
}
```