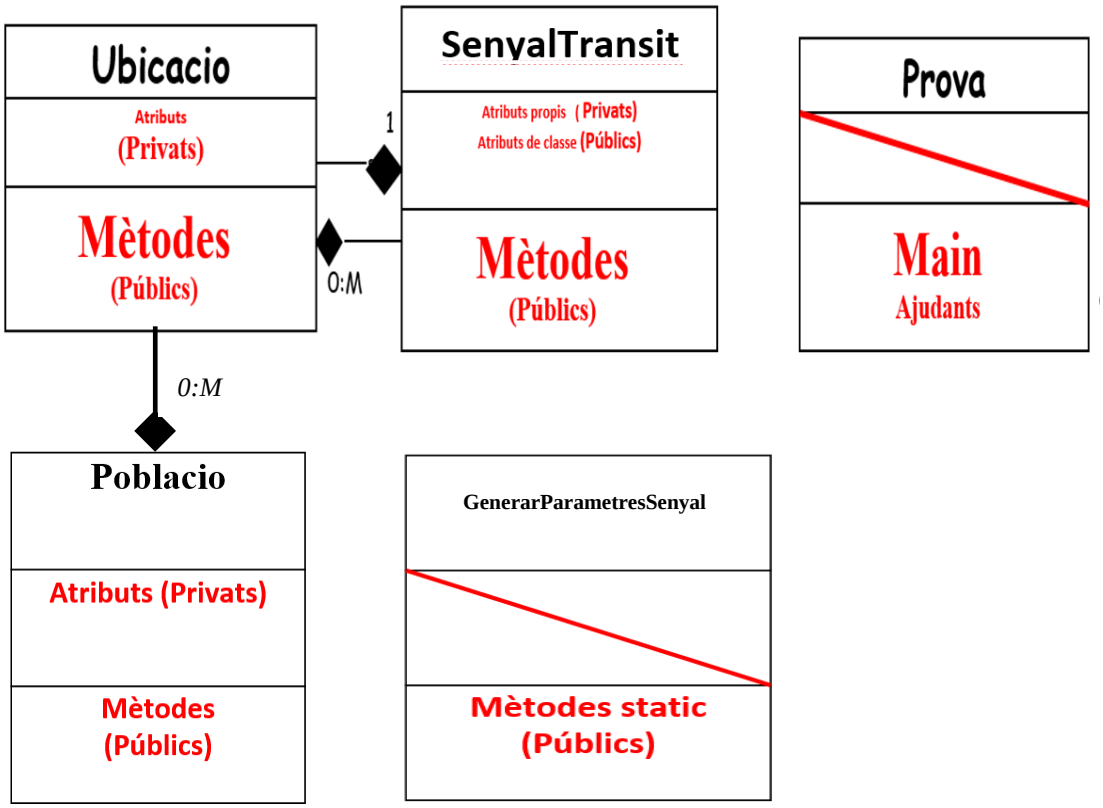
PROGRAMACIÓ ORIENTADA A OBJECTES
TRIMESTRE 2 – CURS 2021/2022
PRÀCTICA 2. SESSIÓ 2

Objectiu: POO. Classes: **this**, atributs i mètodes **static**. **Sobrecarrega** de mètodes
Durada: Dues sessions
Lliurament: Llistat imprès dels fonts i penjar el projecte al Moodle
Data Lliurament: Abans de la Pràctica 3
Important: **Imprescindible per aprovar la pràctica** seguir les pautes de l'enunciat.

Enunciat

Es continua treballant amb el mateix enunciat que la sessió anterior, afegim atributs i mètodes a les classes ja definides i afegim dues noves classes. Una de les classes afegides servirà per generar de manera automàtica algun dels atributs dels objectes SenyalTransit, n'hi direm classe **GenerarParametresSenyal.** Altra classe afegida serà un magatzem de les ubicacions on té senyals la població per a la que s'està fent l'aplicació informàtica, la classe **Poblacio**.

Gràficament



La classe *Ubicacio*

No es modifica res. Es deixa la classe com es demana a la sessió anterior.

La classe SenyalTransit

Sobrecarrega de mètodes constructors. Concretament cal afegir:

- Un constructor amb un **únic paràmetre** corresponent a la ubicació del senyal, mètode que crearà un objecte *SenyalTransit* amb valors aleatoris en tots els seus atributs excepte la ubicació. **Aquests valors aleatoris** seran generats utilitzant els mètodes públics de la ***classe GenerarParametresSenyal***, que es descriurà més endavant. El constructor té la mateixa funcionalitat que el constructor descrit en la sessió prèvia, només que, els valors són generats aleatòriament. Adoneu-vos de què el tipus del senyal l'obtindreu a partir de la primera part del codi del senyal que serà generat aleatòriament.

```
public SenyalTransit(Ubicacio ubicacio)
```

- Altre constructor, en aquest cas es generaran aleatòriament els valors de tots els atributs a excepció de la ubicació i el tipus que vindran donats en els paràmetres. Adoneu-vos que el codi no podrà ser totalment aleatori, la primera part ve donada en el segon paràmetre.

```
public SenyalTransit(Ubicacio ubicacio, int tipus)
```

- Altre constructor, en aquest cas es generaran aleatòriament els valors de tots els atributs a excepció de la ubicació i l'any de col·locació que vindran donats en els paràmetres.

```
public SenyalTransit(int anyColocacio, Ubicacio ubicacio)
```

En la implementació d'aquests constructors és imprescindible que l'un cridi a l'altre amb l'objectiu de reaprofitar codi, useu el **this**.

Sobrecarrega d'altres mètodes. Concretament cal afegir:

- Una sobrecarrega del mètode de la classe SenyalTransit que retira de la via pública un senyal, el que tenim implementat obté l'any de retirada del sistema de l'ordinador, en aquesta sobrecàrrega li vindrà donat en un paràmetre.

```
public boolean  retirarViaPublica (int anyRetirada)
```

Heu de **modificar** la implementació del mètode `boolean  retirarViaPublica()` de la sessió 1 i fer que aquest invoqui a la nova implementació que se us demana.

La classe GenerarParametresSenyal

Com s'ha comentat anteriorment, els constructors afegits a la classe ***SenyalTransit*** utilitza la classe ***GenerarParametresSenyal*** per generar de manera aleatòria els atributs d'un *SenyalTransit*.

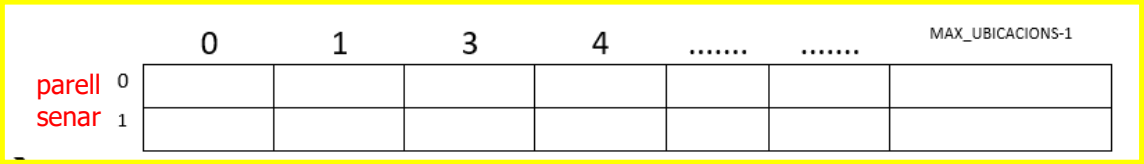
Un aspecte important de ***la classe GenerarParametresSenyal*** és que tots els seus mètodes tenen la signatura ***public static***. Un mètode amb el modificador *static* fa que només existeixi una copia del mètode, de manera que es compartida per tots els objectes de la classe. Així doncs, per invocar als seus mètodes la sintaxi a utilitzar seguirà el següent format: *GenerarParametresSenyal.metodeStatic()*. Teniu més informació a las transparències de teoria.

Els mètodes que ha d'oferir la classe són els següents, heu de seguir les pautes indicades en la seva implementació:

```java
public class GenerarParametresSenyal{
    public static String generarCodi() {
        // Implementeu el mètode que generi un identificador aleatori segons
        // el format "XXX-YYYY" on XXX són lletres i YYYY són números
        // Per fer-ho cal invocar les funcions generarForma i generarDigit
        // El mètode retorna un String que és el codi del senyal
        // COMPLETAR
    }
    public static int generarAny(int maxim) {
      // Implementeu el mètode per generar un valor numèric aleatori de 4
      //xifres compres dins de l'interval [1980, maxim] corresponent a un
      // any
      // COMPLETAR
    }
    public static char generarDigit() {
      // Implementeu el mètode per obtenir un número 0-9 aleatori (van del
      // 48 al 57 en decimal a la taula ASCII)
      // COMPLETAR
    }
    public static String generarForma() {
      // Implementeu el mètode per generar aleatòriament un string de
      // llargària 3 dins del conjunt {"ROD","TRI","REC","QUA"}
      // COMPLETAR
    }
}
```

La classe Poblacio

Un objecte d'aquesta classe és un magatzem amb totes les ubicacions on l'Ajuntament de la població hi té ubicades senyals verticals. Les ubicacions estaran distribuïes en dos magatzems, en un hi hauran les ubicacions que estan en una numeració parell i a l'altre les que estan en una numeració senar. **Aquest magatzem serà una matriu de dues files**, la fila zero emmagatzemarà de forma consecutiva les ubicacions amb numeració parell i la fila u les ubicacions de numeració senar.



Pel que fa als atributs, un objecte Poblacio tindrà els següents atributs privats:

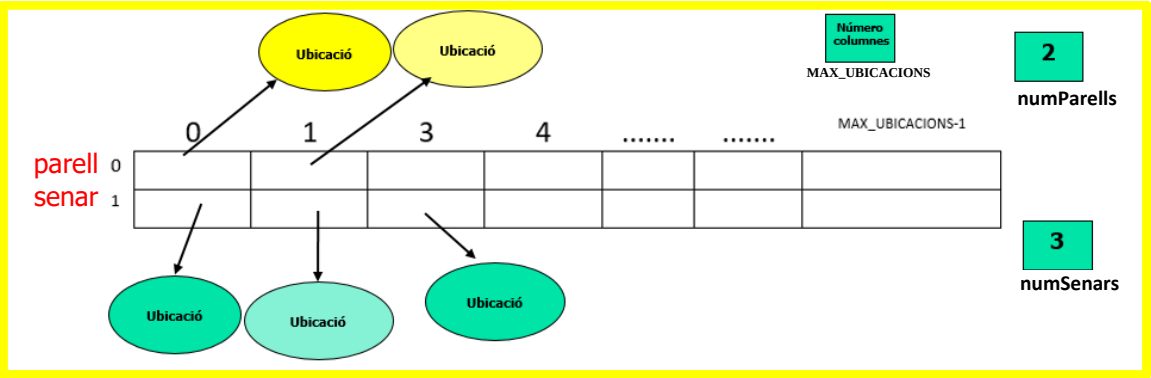
- ***Atribut 1***: que emmagatzema el nom de la població
- ***Atribut 2***: que emmagatzema el nombre d'habitants que té

I finalment altres atributs per emmagatzemar les ubicacions, en concret:

- ***Atribut 3***: s'anomenarà ***ubicacions*** que serà una matriu que servirà de magatzem de les ubicacions que té la població, és a dir contindrà referències a objectes de la ***classe Ubicacio*** (descrita a la sessió 1). **Totes les posicions plenes de cadascuna de les files seran posicions contigües, així doncs les buides estaran també consecutives al final del magatzem**.

- ***Atribut 4:*** s'anomenarà **numParells,** de tipus enter que indicarà el número d'ubicacions que hi ha en aquella població que es troben en numeracions parells. **Emmagatzemarà la dimensió real de la fila d'índex 0 del contenidor ubicacions.**
- ***Atribut 5:*** s'anomenarà **numSenars,** de tipus enter que indicarà el número d'ubicacions que hi ha en aquella població en ubicacions de numeracions senars. **Emmagatzemarà la dimensió real de la fila d'índex 1 del contenidor ubicacions.**
- ***Atribut 6*****:** anomenat **MAX_UBICACIONS** serà un atribut **constant** que s'inicialitzarà en la construcció i no podrà variar, i que representarà el número màxim d'ubicacions que pot haver en la població de cada tipologia (parell, senar). **Emmagatzemarà la dimensió declarada de columnes de la matriu *ubicacions*.**

Exemple: Població amb capacitat per MAX_UBICACIONS ubicacions i que en aquest moment té 3 objectes Ubicacio que no estan en cruïlles i 2 que hi estan.



Pel que fa als mètodes, un objecte Poblacio tindrà els següents mètodes públics:

- Un **mètode constructor** amb els paràmetres corresponents a les dades dels atributs de l'objecte que s'està construint, i que assignarà a cadascun dels atributs mencionats anteriorment. Els atributs corresponents a l'emmagatzematge de les ubicacions s'han d'inicialitzar de la següent manera: la matriu ***ubicacions*** s'ha de crear però ha de romandre buida i el **numParells** i **numSenars** s'han d'inicialitzar a 0 indicant que de moment la població no té ubicacions. La resta d'atributs s'inicialitzaran amb el corresponent paràmetre.

```
public Poblacio(int Maxim, String poblacio, int numHabitants)
//Maxim indica el nombre de columnes de la matriu
```

- Un objecte *Poblacio* té mètodes ***getXXX*** per consultar alguns dels atributs. Concretament:

```
public String getAtribut1();
public int getAtribut2();
public int getAtribut4();
public int getAtribut5();
public int getAtribut6();
```

- **Un *objecte Població* no té mètodes set.**
- El mètode `boolean afegirUbicacio(Ubicacio c)` que rep un objecte de la classe Ubicacio per afegir a la població. S'ha d'afegir si:
  - Hi ha lloc a la matriu, a la fila adient.
  - Si la ubicació a afegir no està ja al magatzem. NO pot haver ubicacions repetides. Dues ubicacions són la mateixa si els tres primers atributs (Atribut 1, 2 i 3 de les ubicacions) coincideixen.

Si es pot afegir la ubicació el mètode retorna true i fals en cas contrari.

- El mètode `boolean eliminarUbicacio(Ubicacio c)` que rep l'objecte de la classe Ubicacio que es vol eliminar de la població. Si es pot donar de baixa la ubicació el mètode retorna cert i fals en cas contrari. No es pot donar de baixa si la ubicació no està al magatzem. Recordeu que no hi poden haver posicions buides entremig.
- El mètode `int eliminarBuides()` que ha de treure del magatzem totes les ubicacions que no contenen cap objecte SenyalTransit. Retorna el número de ubicacions que s'han donat de baixa.
- El mètode `int quantes()` que ha retornar la quantitat de senyals que té la població repartides en les diferents ubicacions.

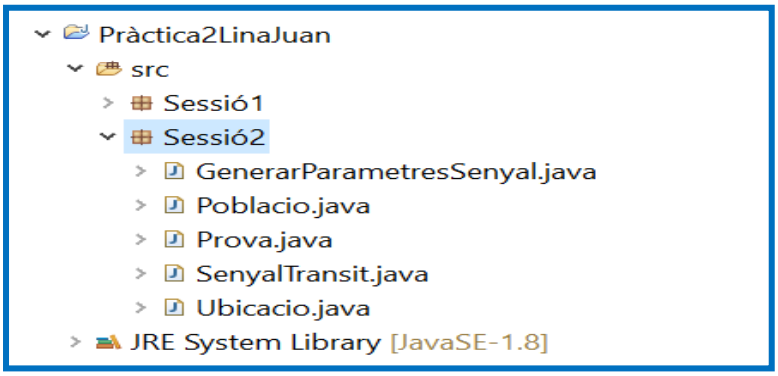
Important: per localitzar una ubicació dins del magatzem només cal buscar-la a la fila adient, depenent de si la ubicació està o no en unanumeració parell o senar. El magatzem ubicacions MAI pot tenir posicions buides entre mig, s'ha de procedir idènticament com s'ha explicat en el magatzem de la classe *Ubicacio* (sessió 1), ara el magatzem és una matriu, però individualment cada fila és un vector.

Tercera part: la classe Prova

A l'igual que a la sessió prèvia el **main** i procediments s'han d'escriure lliurament amb l'objectiu de provar el correcte funcionament de la programació feta en la definició de les classes. No s'haurà de lliurar.

Organització. Totes les pràctiques s'han de fer en grups de 2 estudiants

Seguireu la mateixa estructura que la usada a la pràctica 1. Cal crear un nou projecte amb nom **Pràctica2Cognom1Nom1&&Cognom2Nom2**. El projecte ha de tenir un paquet per cada sessió. En aquesta sessió hi ubicareu les tres classes demanades.



Què se us subministra?

L'enunciat amb les especificacions i línies a seguir per el desenvolupament del disseny de classes.

Què s'ha de lliurar i com?

S'ha de lliurar la carpeta que conté el projecte Eclipse amb el vostre desenvolupament de la pràctica. La carpeta s'ha de lliurar amb tot el seu contingut i comprimida amb ZIP o RAR.

També s'ha de lliurar un **llistat en paper** del codi desenvolupat (no la classe Prova). El format de lliurament d'aquest codi ha de seguir el patró indicat en la presentació de l'assignatura: amb portada, índex, número de pàgina, tabulació ... En **aquest llistat** cal que indiqueu:

- **la distribució de la feina entre els dos estudiants. És a dir, el grau de participació de cada membre del grup en la realització d'aquesta activitat.**

- **Si el programa no funciona cal que indiqueu quina/es parts no funcionen explicant que és el que passa.**

On s'ha de lliurar?

El lliurament del projecte es farà a través de la plataforma Moodle i no s'acceptarà cap altra via. Feu atenció a la data i hora límit.

El lliurament en paper es farà directament a la professora a l'inici de la primera sessió de la pràctica 3.

Quan s'ha de lliurar?

El lliurament es podrà fer fins el **dia indicat a sota**. Tingueu present que a partir d'aquesta hora el sistema bloquejarà, de manera automàtica, la possibilitat de lliurament.

Lliurament Moodle → 24 Febrer a les 23:50h
Lliurament en paper → 25 de Febrer inici sessió de pràctiques

Pauta de correcció:

La qualificació dependrà dels ítems indicats prèviament. **Obligatòriament heu de seguir les pautes de programació** donades en l'escriptura del programa. La vostra solució s'ha d'ajustar a l'enunciat de l'exercici.

Aquesta sessió 2 de la pràctica 2 té un pes del 30% de la qualificació total de la pràctica 2.

Valoració (sobre 10 punts):

- 1.- Escriptura de la classe **Poblacio** (**7 punts**)
- 2.- Escriptura de la classe **GenerarParametresSenyal** (**1.5 punts**)
- 3.- Canvis a la classe **SenyalTransit (1.5 punts)**

La correcció valorarà:

- 1.- **La descomposició funcional** aplicada, tant per un reaprofitament del codi com per la claredat de la solució mostrada. És imprescindible que cada tasca amb una funcionalitat pròpia tingui el seu propi procediment.
- 2.- L'aplicació dels **esquemes** adients (recorregut i cerca)
- 3.- **Eficiència** en quan a no fer càlculs repetits amb unes mateixes dades.

La qualificació dependrà dels ítems indicats prèviament. **Obligatòriament heu de seguir les pautes de programació** donades en l'escriptura del programa. La vostra solució s'ha d'ajustar a l'enunciat de l'exercici.