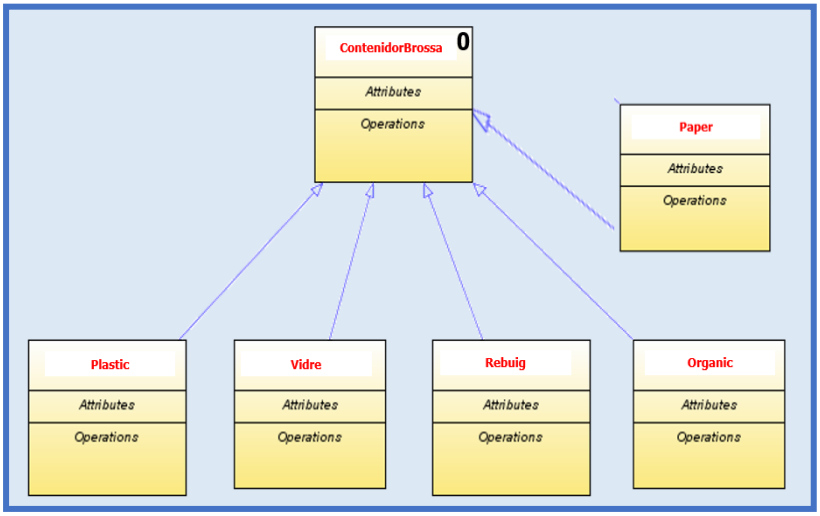
PROGRAMACIÓ AVANÇADA
TRIMESTRE 1. CURS 2022/23
PRÀCTICA 2

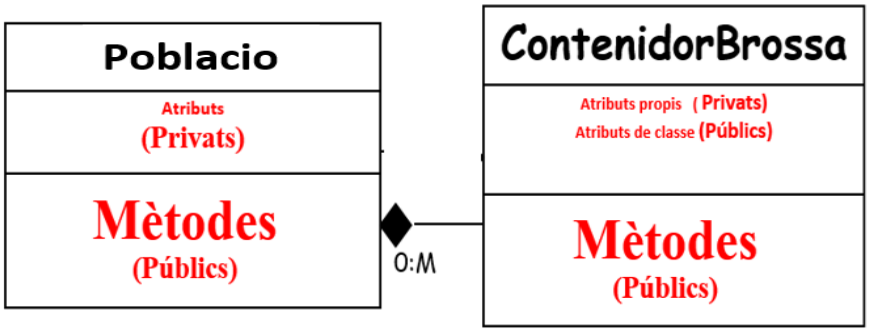
Objectiu 1: Seqüències enllaçades
Objectiu 2: Treballar els conceptes **típics** de la P.O.O.: classes, atributs, mètodes, constructors, destructor, **herència, polimorfisme i redefinició (sobreescriure) de mètodes.**
Objectiu 3: Excepcions.
Durada: Una sessió.
Lliurament: Llistat imprès dels fonts i penjar el projecte al Moodle.
Data Lliurament: El dia previ a la sessió de la pràctica 3.

Enunciat

Cal fer un programa per a controlar els contenidors multicolors de brossa d'un Ajuntament d'una població gran. Com tots sabem el color d'un contenidor indica al ciutadà quin tipus de brossa cal dipositar-hi. A grans trets ens demanen que el software a desenvolupar controli el tipus de contenidor, el seu estat de conservació i la ubicació dins la via pública.



També es vol saber quins són els contenidors que té la població i on estan ubicats.



Primer escriu la classe abstracta ContenidorBrossa

Un contenidor de brossa ve caracteritzat per uns atributs i mètodes que cal implementar, tal com descriurem a continuació.

Pel que fa als atributs de classe, la classe *ContenidorBrossa* tindrà **atributs constants de classe** amb visibilitat pública. Un per a cadascun dels cinc colors possibles dels contenidors, cada atribut estarà codificat mitjançant un valor numèric enter. Colors possibles: groc, marró, verd, gris i blau.

Pel que fa als atributs propis, un objecte *ContenidorBrossa* tindrà els següents atributs propis amb visibilitat privada:

- ***codi***: El codi del contenidor, que en aquest Ajuntament el tenen codificat mitjançant un valor alfanumèric seguint el següent format: "XX-YYYY". XX representen dues lletres corresponents al model del contenidor i YYYY són números que l'identifica. Així doncs un contenidor pot tenir codificacions com ara: "AA-3245" i "TG-3456". Aquest atribut **és identificador**, és a dir, no hi poden haver dos o més contenidors amb el mateix codi.

- ***color***: El color del contenidor ha de ser un valor enter corresponent a alguna de les constants estàtiques de la classe.

- ***ubicacio:*** El nom del carrer on està situat el contenidor. Si té una referència nul.la indica que està al magatzem de l'Ajuntament.

- ***anyColocacio:*** L'any de col·locació, un enter de 4 xifres que representa l'any que entra en servei el contenidor. Si no està ubicat a la via publica el valor d'aquest atribut serà un zero.

- ***anyRetirada:*** L'any de retirada, un enter de 4 xifres que representa l'any que es treu de la via publica el contenidor i per tant està al magatzem de l'Ajuntament. Si encara està ubicat a la via publica el valor d'aquest atribut serà zero.

  Adoneu-vos en referencia als dos darrers atributs, que sempre un dels dos té un valor de zero en qualsevol objecte. Si el contenidor està ubicat al carrer ***l'anyRetirada*** és zero, i si està en el magatzem de l'Ajuntament ***l'anyColocacio*** és el què és zero.

- **tara** d'un contenidor (expressada en Kg). Serà una valor **real** amb visibilitat **protected** amb l'objectiu de tenir-hi accés en les seves classes derivades. La resta d'atributs han de mantenir la seva visibilitat de **private**.

Pel que fa als mètodes, un objecte de la classe *ContenidorBrossa* disposa:

Si el compilador que useu no us permet que un constructor invoqui a l'altre, useu paràmetres per defecte amb un únic mètode constructor

- Un constructor amb **tants paràmetres** com atributs té l'objecte a construir. El mètode crea un objecte *ContenidorBrossa* amb els valors indicats en els paràmetres. Si el valor d'algun paràmetre és incorrecte no es crea l'objecte, cal llançar una excepció. Denoteu que no es dona l'any de retirada perquè amb aquest constructor no s'ubica mai en el magatzem de l'Ajuntament.

```
public ContenidorBrossa(string codi, int color, string ubicacio, int anyColocacio, float tara)
```

- **Sobrecarrega** del constructor. En aquest cas s'ubica el contenidor directament al magatzem de l'Ajuntament, no s'ubica al carrer. Determineu els paràmetres que ha de tenir aquest constructor. **Imprescindible** invocar al constructor previ en la seva implementació.

- El mètode `string getTipusBrossa()` que retorna una cadena informant del tipus de brossa del contenidor. La relació entre el color i el que cal dipositar-hi la trobareu en la imatge prèvia.

- El mètode `void retirarViaPublica()` que simula la retirada de la via pública del contenidor. L'atribut que emmagatzema l'any de retirada s'hi ha d'emmagatzemar l'any actual obtenint-la del sistema.
  Per obtenir la data d'avui (la del sistema de l'ordinador) i obtenir-ne l'any com un enter de 4 xifres, cal procedir de la següent manera (cal incloure la llibreria `#include <time.h>`):

```
time_t now;
struct tm *now_tm;
now = time(NULL);
now_tm = localtime(&now);
int  year= now_tm->tm_year+1900;
```

  La ubicació tindrà una referència nul.la i l'atribut que emmagatzema l'any de col·locació a la via pública ha d'emmagatzemar a partir de la retirada el valor de zero.
  Si el contenidor no està ubicat a la via pública el mètode llança una excepció

- El mètode `string getUbicacio()` retorna el contingut de l'atribut ubicacio. Si està al magatzem ha de llençar una excepció.

- El mètode `void setUbicacio(string ubicacio)` per modificar l'atribut ubicació. Cal tenir present que si el paràmetre és nul serà una retirada de la via pública, cal invocar al mètode corresponent.

- El mètode `int getCodi()` que retorna el valor de l'atribut codi del contenidor.

- El mètode `string getEstat()` que retorna una cadena amb l'estat del contenidor de brossa, seguint la següent pauta:

  - Si té més de 5 anys retorna la cadena vell
  - Si té entre 3 i 5 anys retorna seminou.
  - Si té menys de 3 any retorna nou.

  Els anys que té un contenidor ve donat per la diferencia entre l'any de la data a dia d'avui i l'any de col·locació. Si el contenidor no està a la via pública el mètode ha de retornar la cadena "retirat".

- **Sobrecàrrega dels operadors: < > ==** La relació d'ordre s'estableix en funció del codi dels contenidors que es comparen.

```
bool operator==(ContenidorBrossa *p);
bool operator<(ContenidorBrossa *p);
bool operator>(ContenidorBrossa *p);
```

Observeu que el paràmetre és un punter.
Obligatòriament en la implementació d'un d'aquest operadors s'ha d'invocar als altres **dos, i a més usant directament la simbologia de l'operador**. **Recomanació**: usar el mètode `compare` per comparar string (llibreria <string.h>)

- Implementa el mètode **`void toString()`** que ha de visualitzar a pantalla l'objecte mostrant el contingut de tots els seus atributs.

Exemple:

```
Codi: XX-YYYY
Color: en format string, invoca al mètode adient
Ubicació: nom de carrer si està en servei, "retirat" si està al
magatzem de l'Ajuntament
Tara: valor real amb dos decimals
```

- Aprofitant que els vehicles que buiden els contenidors porten un sensor per a mesurar el pes del contenidor de brossa, cal afegir el mètode abstracte:

```
void buidat (float pes)
```

que rebrà com a paràmetre el pes en quilograms del contenidor abans de buidar-lo. Aquest mètode serà implementat per cadascuna de les subclasses que representaran els diferents tipus de contenidors de brossa.

Una conseqüència d'afegir aquest mètode és que provoca que la classe ContenidorBrossa es converteix en una classe **abstracta**. I per tant, no se'n podran crear objectes.

- El mètode **`string getType()`** que ha de ser abstracte. Les classes derivades l'han de redefinir per retornar la cadena corresponent al nom de la classe. La funcionalitat que se li dona a aquest mètode és per saber de quina classe és l'objecte sobre el que s'invoca. Concretament és per "simular" l'operador **instanceof** de Java que el C++ no té.

- Cal afegir altre mètode abstracte: **`string getReciclat ()`**. Aquest mètode serà implementat per cadascuna de les subclasses que representaran els diferents tipus de contenidors de brossa. Retornarà la quantitat de material reciclat pel contenidor.

- El mètode **destructor** (té cos buit però necessari que hi sigui).

```
~ContenidorBrossa();
```

Segon escriu les classes derivades: les subclasses Organic, Rebuig, Vidre, Plàstic i Paper

Crear les classes **Organic**, **Rebuig**, **Vidre**, **Plastic** i **Paper** com a subclasses de la classe **ContenidorBrossa**.
L'herència entre classes ha de mantenir la mateixa visibilitat a l'heretada, és a dir, cal fer una **herència pública**.

Afegir a cada classe:

- ✓ Un atribut privat, anomanat ***reciclat***, que segons el contenidor serà del tipus:
  - **real (float)** pels orgànics i rebuig que representarà les **tones** de material reciclat.

- **real (float)** pels de paper i plàstic que representarà els **quilograms** de material reciclat.
- **enter llarg** (**long**) pels de vidre que representarà el **nombre** d'envasos equivalents reciclats.

✓ Un constructor que rebi com a paràmetres, el codi, la ubicació, l'any de col·locació i la tara del contenidor. **Cada subclasse informarà a la superclasse del seu color (imprescindible usar els atributs estàtics de la classe ContenidorBrossa). No ha de ser un paràmetre del constructor. L'atribut propi s'inicialitza a zero.**

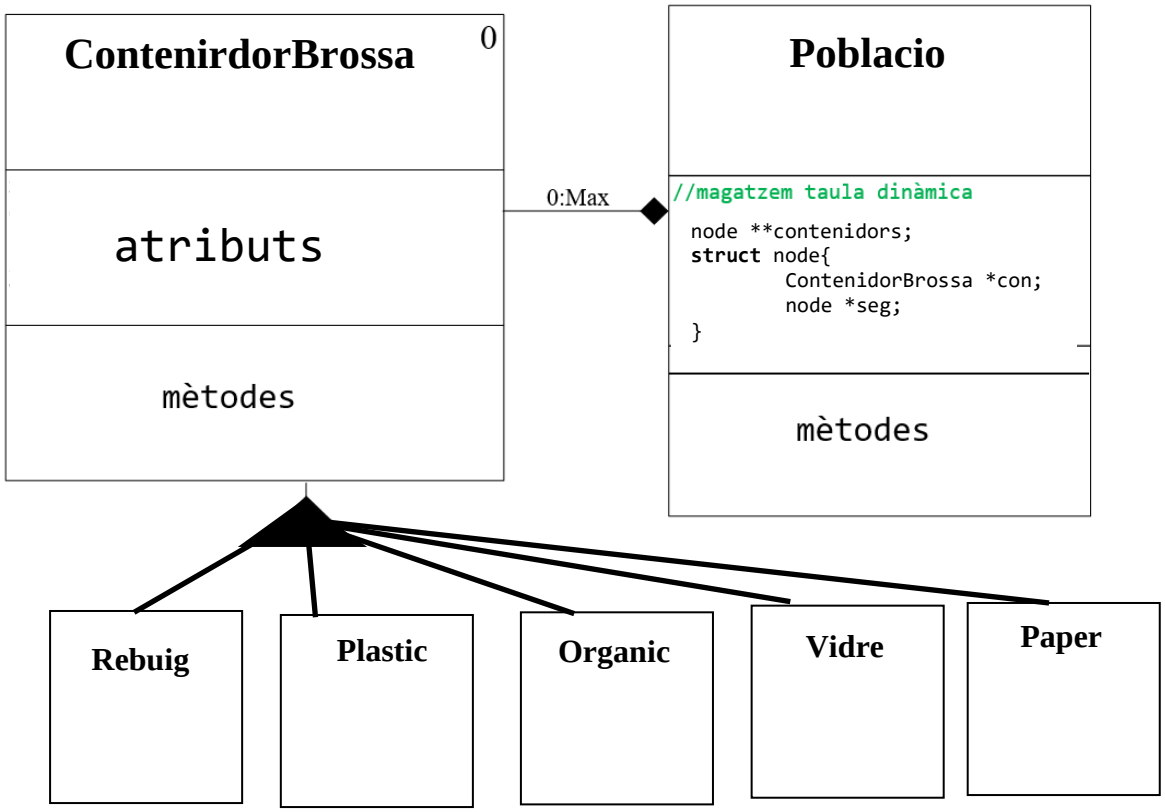
✓ **Redefineix el mètode consultor** `getReciclat()` per obtenir la quantitat de material reciclat pel contenidor. Aquest mètode ho retorna en un string que, a més del valor numèric de l'atribut propi, adjunti la unitat de mesura: quilograms, tones o nombre d'unitats depenent del tipus de contenidor.

✓ **Redefineix el mètode toString()** que ha d'invocar al corresponent de la superclasse i afegir-li el contingut de l'atribut propi afegit invocant al mètode get previ.
✓ **Mètode** `string getType()` cal implementar-lo per a que retorni la cadena corresponent al nom de la classe.
✓ **Mètode destructor** (té cos buit però necessari que hi sigui).
✓ Implementar el mètode abstracte heretat `public void buidat (float pes)` que actualitzarà l'atribut propi de cada classe, segons el tipus de contenidor, en tots els casos caldrà calcular els quilograms de material reciclat, que serà la diferència entre els quilograms rebuts com a paràmetre i la tara de cada contenidor. En funció del tipus de contenidor:

- **Orgànic**, un cop calculat el pes en quilograms del material del contenidor buidat, caldrà calcular les tones reciclades sabent que per cada quilogram només el 90% és efectiu. Finalment caldrà sumar aquestes tones a l'atribut propi.
- **Rebuig**, igual que el cas de l'orgànic, però essent l'efectivitat del 75% per cada quilogram de material reciclat.
- **Plàstic**, en aquest cas l'efectivitat és del 80% per quilogram reciclat, però tenint en compte que l'atribut propi és en quilograms.
- **Paper**, igual que els contenidors de plàstic, amb una efectivitat 95%.
- **Vidre**, cada quilogram de vidre reciclat es podran generar tres envasos nous. Caldrà sumar la nova quantitat que resulti del buidat a l'atribut propi.

Tercer escriu la classe Poblacio

Un objecte Poblacio és un magatzem de contenidors de brossa. El magatzem és un vector de cinc components, una per cada color de contenidor, i cada component és una referència a una seqüència enllaçada sense node capçalera d'objectes *ContenidorBrossa* que són del mateix color.



Atributs:

Vector **dinàmic** de 5 components. Cada component és un punter a un node.

```
node** contenidor;
```

El tipus node és un **struct** amb els dos atributs habituals: un per emmagatzemar la informació i altre per referenciar al següent node i així crear la seqüència enllaçada.

```
struct node{
        ContenidorBrossa *con;
        node* seg;
};
```

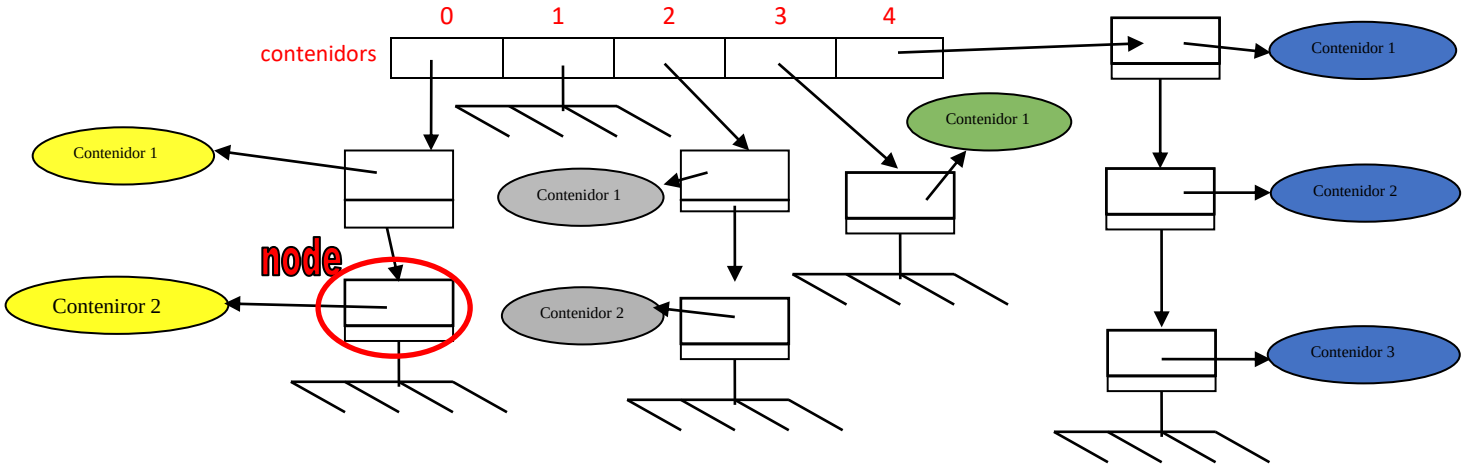
La informació emmagatzemada en cada node, tant en el camp *con* com en el *seg* són punters a un objecte creat dinàmicament. **Tots els nodes que formin les seqüències enllaçades s'han de crear dinàmicament.**

Per **crear un vector dinàmicament** s'ha de seguir les següents passes:

```
tipus* variable;
variable = new tipus[dimensio];
//ús de la variable amb la sintaxi de vector variable[índex]
....
delete variable; //imprescindible
```

Tant l'atribut com el tipus node han de tenir una visibilitat privada.

Gràficament. Exemple:



En el gràfic anterior, la població, no conté contenidors de rebuig, és a dir de color marró. La seqüència enllaçada referenciada per la posició d'índex **zero** del vector conté tots els contenidors de color groc, la **u** els marró, la **dos** els de color gris, la **tres** els verds, la **quatre** els blaus.

Aquesta classe no té membres de classe. Sí que té un nou tipus de dades, de visibilitat privada, corresponent al node.

NOTA: no tenim cap mètode per poder consultar el color del contenidor, i en C++ no tenim l'operador `instanceof`, en conseqüència per saber quina seqüència enllaçada li correspon a un contenidor invocareu obligatòriament al mètode `getType()`.

Mètodes:

- **Mètode constructor** sense paràmetres. Ha de crear el vector i reflectir que les seqüències enllaçades no tenen nodes.
- **Mètode constructor** sobrecarregat, aquest amb un paràmetre: un objecte dinàmic ContenidroBrossa per afegir-lo a la Poblacio un cop creada. Aquest mètode **obligatòriament** ha d'invocar al constructor previ i al mètode que afegeix un contenidor a la població (explicat més endavant). Suposeu que el valor del paràmetre és correcte.

```
Poblacio(ContenidroBrossa *g);
```

- Mètode **`void afegirContenidor(ContenidorBrossa *p)`** que simula l'alta **d'un** contenidor. El mètode llança una excepció si el contenidor ja està al magatzem. Dos contenidors són iguals si tenen el mateix codi, **invoqueu obligatòriament l'operador d'igualtat sobrecarregat de la classe ContenirBrossa. Sempre que es tingui que localitzar un contenidor brossa obligatòriament cal invocar a l'operador == usant la simbologia de l'operador.**
- **Sobrecarrega** del mètode previ, **`void afegirContenidor(string codi, int color, string ubicacio, int anyColocacio, float tara)`** que simula l'adquisició d'un contenidor, però en aquest cas no es dona l'objecte ContenidroBrossa, sinó què arriben els paràmetres necessaris per la seva construcció. **Aquest mètode obligatòriament ha d'invocar al precedent**.
- Mètode **`string hiEs(string codi)`** ha de retornar el nom del color del contenidor brossa que té per codi el donat en el paràmetre. Si el contenidor no és de la població ha de llançar una excepció.

- Un mètode **`void eliminarContenidor(ContenidorBrossa *c)`** per eliminar del magatzem el contenidor especificat en el paràmetre. Si no hi és cal llançar una excepció.
- Mètode **`ContenidorBrossa* mesRendiment()`** que ha de trobar el contenidor de la població que té més rendiment (recicla més). En cas d'igualtat és irrellevant el retornat. Llança una excepció si la població no té cap contenidor.
- Mètode **`int getQuants(int color)`** per calcular quants contenidors té la població del color indicat en el paràmetre.
- Mètode **`int getQuants()`** sobrecarrega del precedent, en aquest cas ha de calcular el nombre **total** de contenidors que té la població, independentment del seu color. **Obligatòriament** ha d'invocar al mètode precedent.
- **Sobrecàrrega dels operadors: < > ==** La relació d'ordre s'estableix en funció del número total de contenidors que té la població, com més en té més gran és. **El paràmetre d'aquestes sobrecarregues no és dinàmic.**

```
bool operator==(Poblacio d);
bool operator<(Poblacio d);
bool operator>(Poblacio d);
```

Obligatòriament en la implementació d'un d'aquest operadors s'ha d'invocar als altres dos, i a més usant directament la simbologia de l'operador.

- El mètode **`void toString()`** per visualitzar a pantalla els atributs de tots els contenidors que té la població. Ha de seguir el següent format:

```
Contenidors grocs
Visualització a pantalla del contenidor 1
Visualització a pantalla del contenidor 2
....
Contenidors marrons
Visualització a pantalla del contenidor 1
Visualització a pantalla del contenidor  2
....
Seguir aquest patró per tots els colors. Irrellevant l'ordre dels colors
```

Per visualitzar els atributs del contenidor invoqueu al mètode **toString()** sobre cada contenidor

- **destructor imprescindible** per alliberar l'espai ocupat per la població: contenidors, vector i nodes.

PROVA:

Per provar el correcte funcionament cal que escriguis un main per comprovar-ho. No és necessari el seu lliurament.

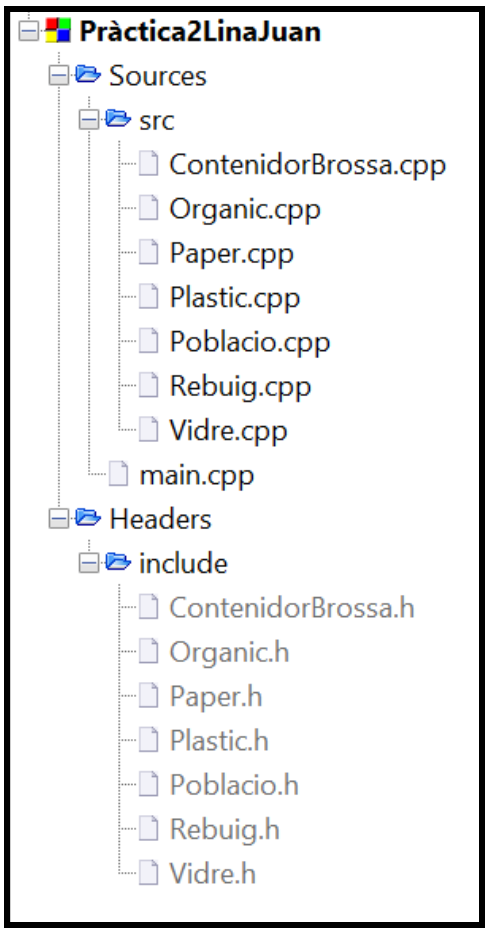
Aclariment

Es valorarà l'eficiència de les implementacions. Per una banda, si amb un únic recorregut i/o cerca es pot determinar el que ha de calcular el mètode no se'n faran ni dos, ni més. **Per altra banda**, s'ha d'aplicar descomposició funcional en la implementació d'aquells mètodes que poden reaprofitar codi o senzillament són llargs i es trossegen per millorar la seva bona entesa.

Organització

Cadascuna de les pràctiques que desenvolupareu en aquesta assignatura ha de ser un projecte diferent. Les pràctiques obligatòriament s'han de fer en grups de 2 estudiants.

Els noms dels vostres projectes, en les pràctiques d'aquesta assignatura han de seguir el següent patró: PràcticaXCognom1Nom1&&Cognom2Nom2. En aquesta pràctica X és 2.



Cal tenir separadament els fitxes de capçalera dels fitxers que tenen les implementacions de les classes

Què se us subministra?

→Fitxer amb l'enunciat de la pràctica. Document que esteu llegint.
→Document: Apunts de Programació: C++. Part 1 i Part 2.
→Document: Excepcions en C++.
→Document: Variable dinàmica en C++.

Què s'ha de lliurar i com?

S'ha de lliurar la carpeta que conté el **projecte** Code::Blocks 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 el **llistat en paper** del codi desenvolupat. 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 quina ha estat **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. Explica les principals dificultats en el seu desenvolupament, valoració del codi lliurat i funcionament de la pràctica.**

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 pràctica 3.

Quan s'ha de lliurar?

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

Tots els grups → 2 de Novembre a les 8h

Pautes de Correcció

La pràctica serà puntuada sobre 10. Concretament la distribució de punts es farà de la següent forma:

Classe ContenidroBrossa → 1.5 punts
Classes derivades → 1 punt
Classe Poblacio → 7.5 punts

Es valorarà l'eficiència de les implementacions. Per una banda, si amb un únic recorregut i/o cerca es pot determinar el que ha de calcular el mètode no se'n faran ni dos, ni més. **Per altra banda**, s'ha d'aplicar descomposició funcional en la implementació d'aquells mètodes que poden reaprofitar codi o senzillament són llargs i es trossegen per millorar la seva bona entesa. **Imprescindible seguir les pautes indicades a l'enunciat.**