Laboratori de Programació

Guió de la 1ra Part Projecte 2023-24

Objectius

1. Implementar una solució de programari a través dels conceptes d'herència i d´estructures de dades dinàmiques de manera eficient.
2. Comprendre una base de software que se'ns ofereix per tal d'implementar noves funcionalitats.
3. Implementar els algoritmes i estructures de dades necessaris per la cerca del camí més curt entre dos punts especificats.

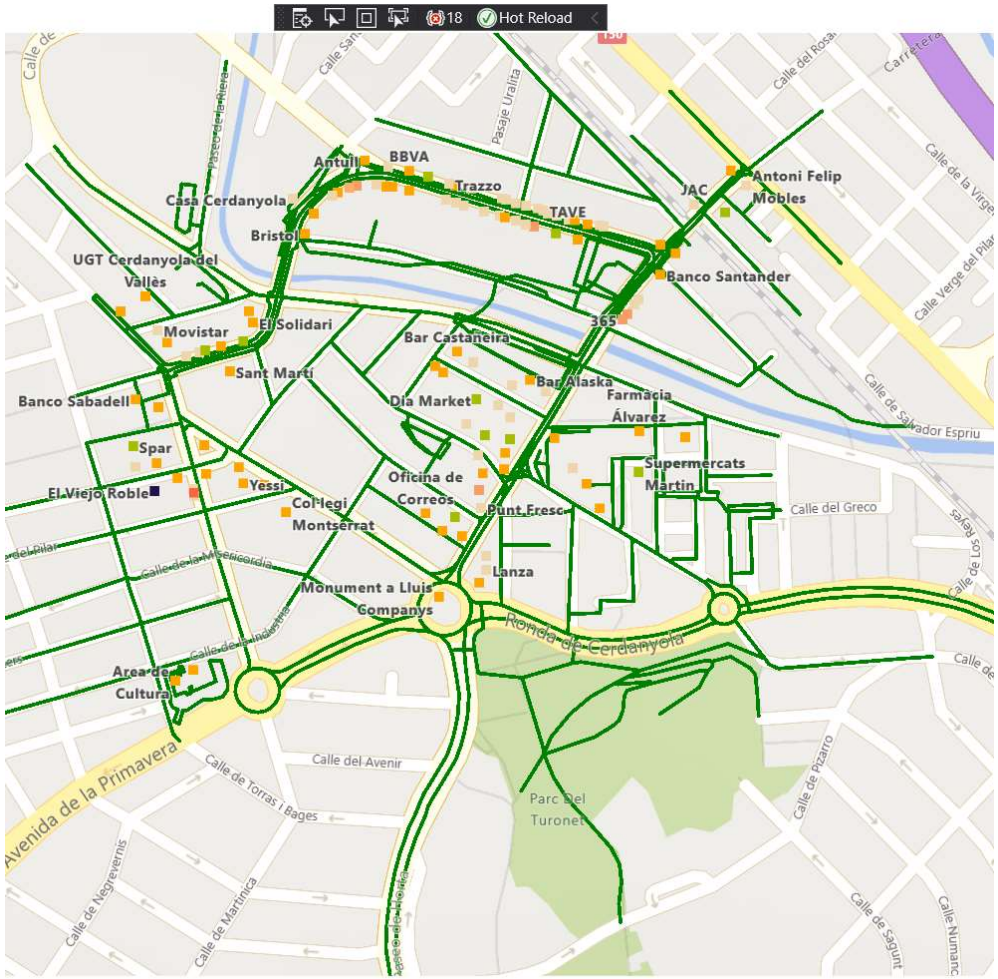
Parts del Projecte

Primera Part

La primera part consistirà en posar en funcionament l'entorn gràfic donat, i familiaritzar-vos amb l'avaluador de Caronte. A més de llegir el fitxer XML amb la informació, per desar-la en l'aplicació. Per fer-ho primer haureu de dissenyar i implementar l'estructura de dades més adient per crear, guardar i accedir a: Nodes de camins, camins i punts d'interès. Per després mostrar-los a un mapa.

Entorn de desenvolupament: Visual Studio vs 2019, compilació C++14

https://visualstudio.microsoft.com/es/thank-you-downloading-visual-studio/?sku=Community&rel=16



Projectes a Caronte

PER PRACTICAR I ENTENDRE LA INFORMACIÓ QUE TENIU

- **codiEstudiantsTasca1-2Fitxer:** En aquest projecte realitzareu les tasques 1 i 2. Us serveix principalment per conèixer com OSM estructura les dades. Realitzant les tasques que se us demana, us facilitarà entendre les tasques posteriors. No s'ha de lliurar.

- **codiGràficEstudiantsPart1:** Aquest projecte us servirà per comprovar que els punts i camins que esteu definint són els correctes de manera visual. Podeu començar a realitzar les tasques aquí.

EL PROJECTE PER LLIURAR

- **Avaluador Projecte – Part 1 – Grup XXX:** Entrega a Caronte a través del vostre grup del projecte. Aquest és el codi responsable de la qualificació, que es realitza de manera automàtica.

Tasca 1 i Tasca 2

Aquestes tasques serveixen per familiaritzar-vos amb l'entorn que hem creat.

Primer us heu descarregat el projecte de Caronte: codiEstudiantsTasca1-2Fitxer

Alla teniu un projecte creat i dos **XmlElement**: (XmlElement definit a **Common.h)**

- Un que representa un node d'interès: viejoRoble
- Un que representa un camí: camiJoaquim

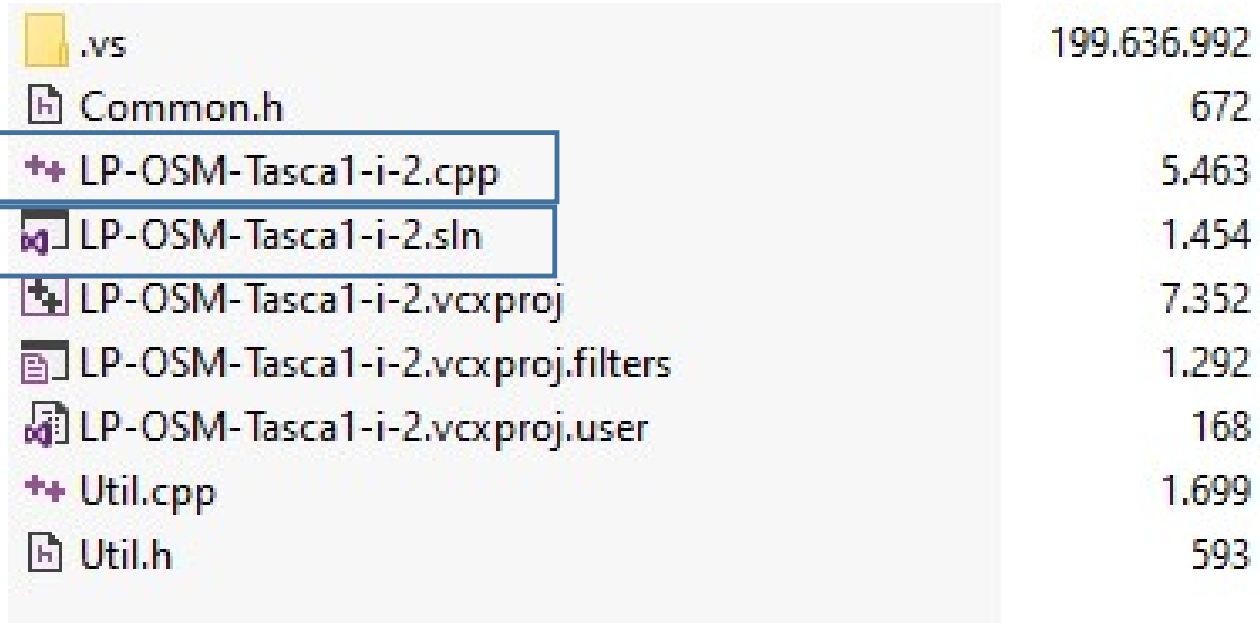
Tasca 1 i Tasca 2

Aquestes tasques serveixen per familiaritzar-vos amb l'entorn que hem creat.

Primer us heu descarregat de Caronte: **`codiEstudiantsTasca1-2Fitxer`**

Allà teniu un projecte creat i dos XmlElement:

- Un que representa un node d'interès: **viejoRoble**
- Un que representa un camí: **camiJoaquim**



Tasca 1

```
XmlElement viejoRoble = {
      //id_element =
      "node",
      // atributs =
      {
            std::make_pair("id", "357101444"),
            std::make_pair("visible", "true"),
            // ... Codi restant. Veure complert a estudiants-tasca1-2
            std::make_pair("uid", "4946691"),
            std::make_pair("lat", "41.4902204"),
            std::make_pair("lon", "2.1406477")
      },
      // fills =
      {
            std::make_pair<std::string,std::vector<std::pair<std::string, std::string>>>
            ("tag", {
            std::make_pair<std::string, std::string>("k", "amenity"),
            std::make_pair<std::string, std::string>("v", "restaurant")
            }),
      // ... Codi restant. Veure complert a estudiants-tasca1-2
      }
};
```

```
/*Tasca 1:
Imprimir per pantalla ident, latitud (lat),
l'longitud (lon) i nom (name) de l'element
XML viejoRoble
Transformar latitud i longitud a double.
A la pràctica haureu de recollir tots els
punts d'interés (nodes que no són nodes de
camí que tenen un atribut name)*/

// TODO Tasca1
```

Iterar **viejoRoble** per tenir id, lat i lon, per Tasca 1

Tasca 2

Per **camiJoaquim**, a Tasca 2, imprimir els identificadors dels nodes que el formen

Tasca 2:
Imprimir identificadors dels nodes que formen XmlElement camiJoaquim, i valor atribut highway.

A la pràctica només tots els camins amb tag highway i saber quins nodes de tipus camí, formen cada camí.

```
XmlElement camiJoaquim = {
// id_element ="way",
// atributs =
{std::make_pair("id", "31889094"),
 std::make_pair("visible", "true"),
 std::make_pair("version", "6"),
//…
},
// fills =
{std::make_pair<std::string, std::vector<std::pair<std::string, std::string>>>
("nd",{std::make_pair("ref", "357125056")}),
//…
std::make_pair<std::string, std::vector<std::pair<std::string, std::string>>>
("tag", {std::make_pair("k", "highway"),std::make_pair("v", "living_street")}),
std::make_pair<std::string, std::vector<std::pair<std::string, std::string>>>
("tag", {std::make_pair("k", "name"),std::make_pair("v", "Carrer de Sant Joaquim")}),
//…
};
```

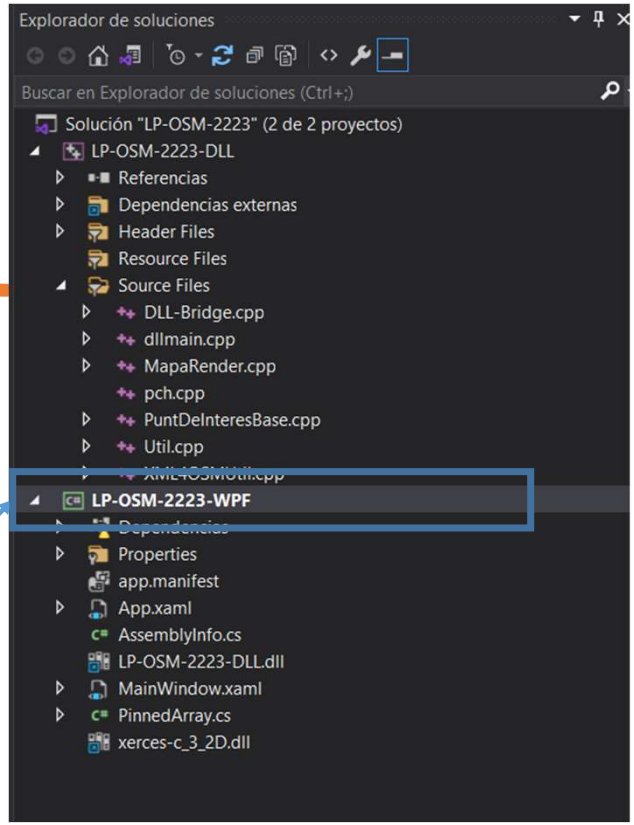
```
valorTag = Util::kvDeTag()//valorTag.first ‘highway’; valorTag.second ‘living_street’
```

Interfície Gràfica

Allà teniu dos projectes dins de la solució de Visual Studio:

- LP-OSM-2223-DLL: Presenta la part de càlcul i estructures de dades.
  - Està escrit en C++, i és el projecte on haureu d'implementar la vostra solució.
  - Aquest guió pretén explicar exclusivament les classes definides en aquest projecte.

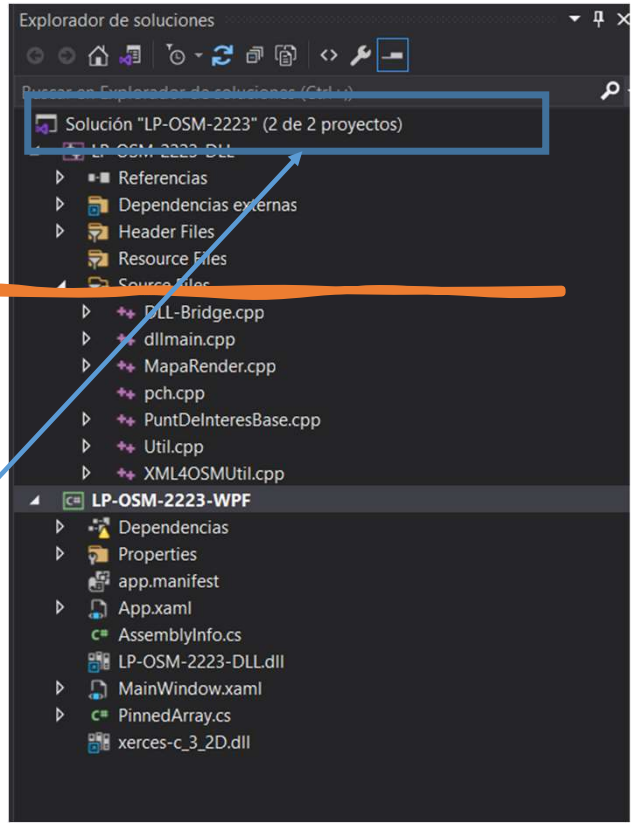
- LP-OSM-2223-WPF: Presenta l'entorn visual, i està escrit en C# WPF.
  - **NO heu de modificar res de cap fitxer d'aquest projecte per assòlir els objectius de la pràcticac**



Tasques 3-6 Projecte

Aquestes tasques serveixen per començar a definir classes del projecte i mirar com funcionen

Primer us heu descarregat el projecte de Caronte: codiGraficEstudiants



Allà teniu dos projectes:

- LP-OSM-2223-DLL: Presenta la part de càlcul i estructures de dades.
  - Està escrit en C++, i és el projecte on haureu d'implementar la vostra solució.
  - Aquest guió pretén explicar exclusivament les classes definides en aquest projecte.

- LP-OSM-2223-WPF: Presenta l'entorn visual, i està escrit en C# WPF.
  - **NO heu de modificar res de cap fitxer d'aquest projecte per asolir els objectius de la pràctica**

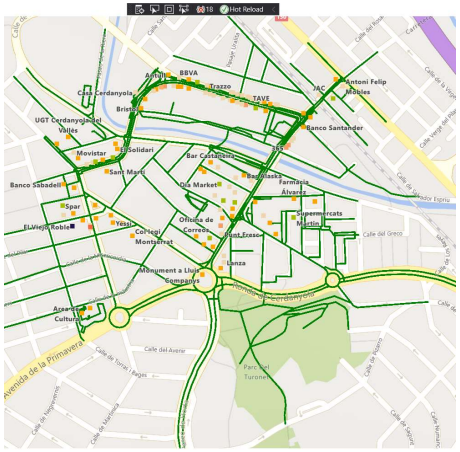
Flux tècnic transparent

LP-OSM-2223-DLL

Genera una DLL, que crida els mètodes que implementeu.

Importa la DLL, transferint les dades que retorneu

LP-OSM-2223-WPF



Representació gràfica

Per això és **important** que utilitzeu **l'herència i implementeu els mètodes que us demanem**!

Altra manera, els dos projectes no s'entendràn correctament!

Tasca3: PuntDeInteresBase

Tenim diferents punts d'interès, però la classe base ens indica una interface comuna a tots:

```
class PuntDeInteresBase {
    private:
        Coordinate m_coord;
        std::string m_name;

    public:
        PuntDeInteresBase();
        PuntDeInteresBase(Coordinate coord, std::string name);

        virtual std::string getName();
        Coordinate getCoord();
        virtual unsigned int getColor();
};
```

Crear classes derivades **PuntDeInteresBotigaSolucio** i **PuntDeInteresRestaurantSolucio**

Implementar mètodes a les classes derivades, segons als atributs descrits a l'enunciat

Coordinate: Estructura amb dos camps: lat i lon, que defineix la posició en el mapa. Definida a Common.h

```
typedef struct {
    double lat;
    double lon;
} Coordinate;
```

Tasca3: PuntDeInteresBotigaSolucio

```
class PuntDeInteresBase {
    private:
        Coordinate m_coord;
        std::string m_name;

    public:
        PuntDeInteresBase();
        PuntDeInteresBase(Coordinate coord,
                          std::string name);

        virtual std::string getName();
        Coordinate getCoord();
        virtual unsigned int getColor();
};
```

Afegir 3 atributs:

- El valor del tag “shop”
- El valor del tag "opening_hours"
- El valor de "wheelchair"

Retornarà:

1. 0xA5BE00 pels “supermarket”
2. 0xFFAD69 pels “tobacco”
3. 0xE85D75 pels “bakery”.
4. 0x4CB944 pels “bakery” amb “opening_hours” que contingui el text “06:00-22:00” amb "wheelchair" igual a "yes" (accés per persones amb mobilitat reduïda).
5. 0xEFD6AC en els altres casos (altres “shop”)

definició d’altres colors, podeu utilitzar https://coolors.co/generate

Tasca3: PuntDeInteresRestaurantSolucio

```
class PuntDeInteresBase {
    private:
        Coordinate m_coord;
        std::string m_name;

    public:
        PuntDeInteresBase();
        PuntDeInteresBase(Coordinate coord, std::string name);

        virtual std::string getName();
        Coordinate getCoord();
        virtual unsigned int getColor();
};
```

Dos atributs addicionals:

- Tipus de cuina (“cuisine”)
- Si facilita accés a persones amb mobilitat reduïda

Retornarà:

1. **0x03FCBA** pels “pizza”+ mobilitat reduïda
2. **0xA6D9F7** pels “chinese”
3. **0x251351** per la resta amb mobilitat reduïda ("wheelchair").
4. **Color definit per defecte** en els altres casos

Nota: Tingueu en compte que el color per defecte ve definit pel PuntDeInteresBase, i pot canviar.

Tasca 4: CamiBase

Volem tenir definits els camins

```
#include "Common.h"
#include <vector>

class CamiBase {
    public:
        virtual std::vector<Coordinate> getCamiCoords() = 0;
};
```

Crear classe
Derivada
CamiSolucio

Implementar mètode, i definir atributs i mètodes
extra a la clase derivada
segons necessàri
per afegir les coordenades per **MapaSolucio**

Tasca 4: CamiSolucio

Volem tenir definits els camins

```
#include "Common.h"
#include <vector>

class CamiBase {
    public:
        virtual std::vector<Coordinate> getCamiCoords() = 0;
};
```

Crear clase Derivada **CamiSolucio**

Implementa mètode getCamiCoords()

`Coordinate`: Estructura amb dos camps: `lat` i `lon`, que defineix la posició en el mapa d'un node del camí.

Tasca 4: Prova aïllada per visualitzar un primer camí

Que getCamiCoords() retorni el següent conjunt de coordenades, definit per (lat, lon)

- (41.4928803, 2.1452381)
- (41.4929072, 2.1452474)
- (41.4933070, 2.1453852)
- (41.4939882, 2.1456419)

Tasca 5 Tasca 6: MapaSolucio

```
class MapaBase {

    public:
        virtual void getPdis(std::vector<PuntDeInteresBase*>&) = 0;

        virtual void getCamins(std::vector<CamiBase*>&) = 0;

        virtual void parsejaXmlElements(std::vector<XmlElement>&
                xmlElements) = 0;

};
```

Crear clase derivada **MapaSolucio**

Agafar les dades de xmlElements per obtenir Punts de Interes i Camins, a MapaSolucio

Retornar tot el conjunt de camins parsejats, a MapaSolucio

Retornar tot el conjunt de Punts d'Interes parsejats, a MapaSolucio

Tasca 5: MapaSolucio

```
class MapaBase {
    public:
        virtual void getPdis(std::vector<PuntDeInteresBase*>&) = 0;
        virtual void getCamins(std::vector<CamiBase*>&) = 0;
        virtual void parsejaXmlElements(std::vector<XmlElement>& xmlElements) = 0;
};
```

Crear clase derivada **MapaSolucio**

Vosaltres haureu de crear una classe derivada de **MapaBase**, que es dirà **MapaSolucio** i que guardarà la seva informació com a estructures internes. Aquestes estructures les haureu de pensar, dissenyar i implementar perquè siguin òptimes, i guardaran la informació del fitxer XML (punts d'interès i camins formats per nodes).

IMPORTANT: En utilitzar l'interfície gràfica, el metodes són cridats mitjançant les funcionalitats de l'interfície gràfica. Es recomana SEMPRE utilitzar Fitxer -> Obrir, càrregar un fitxer, i llavors, utilitzar els botons per visualitzar l'informació corresponent.

Tasca 5: MapaSolucio

```
class MapaBase {
    public:
        virtual void getPdis(std::vector<PuntDeInteresBase*>&) = 0;
        virtual void getCamins(std::vector<CamiBase*>&) = 0;
        virtual void parsejaXmlElements(std::vector<XmlElement>& xmlElements) = 0;
};
```

Tasca 5: Implementeu el mètode getPdis(std::vector<PuntDeInteresBase*>&)

1. Retorna dos punts d'interès (que heu implementat a la **Tasca 3**):

- un **PuntDeInteresBotigaSolucio:** ***shop*** amb valor "***bakery***" i nom "La Millor Pastisseria" situada a (41.4918606, 2.1465411)
- un **PuntDeInteresRestaurantSolucio**, ***amenity*** amb valor ***restaurant,*** cuina ***regional*** i mobilitat reduïda. Situat a (41.4902204, 2.1406477), nom "El Millor Restaurant".

```
MapaRender::MapaRender()
{this->m_xml4osm_util = new XML4OSMUtil();
 // TODO: instancieu m_mapaBase amb la vostra MapaSolucio
 this->m_mapaBase = nullptr;
}
```

Nota: Instancieu correctament la propietat m_mapaBase de la classe **MapaRender** construint la vostra solució de **MapaSolució**

Tasca 6: MapaSolucio

```
class MapaBase {
    public:
        virtual void getPdis(std::vector<PuntDeInteresBase*>&) = 0;
        virtual void getCamins(std::vector<CamiBase*>&) = 0;
        virtual void parsejaXmlElements(std::vector<XmlElement>& xmlElements) = 0;
};
```

Tasca 6: Retorneu el camí que heu implementat a la **Tasca 4** mitjançant l'implementació de getCamins(std::vector<CamiBase*>&)

- **`void getCamins(std::vector<CamiBase*>& camins)`**
  - Retorna tots els camins que haureu parsejat i que són **highway**.
  - Nota: La **classe CamiBase** és una classe virtual pura i necessitareu definir una classe derivada **`CamiSolució`** on afegireu nous atributs i funcionalitats.

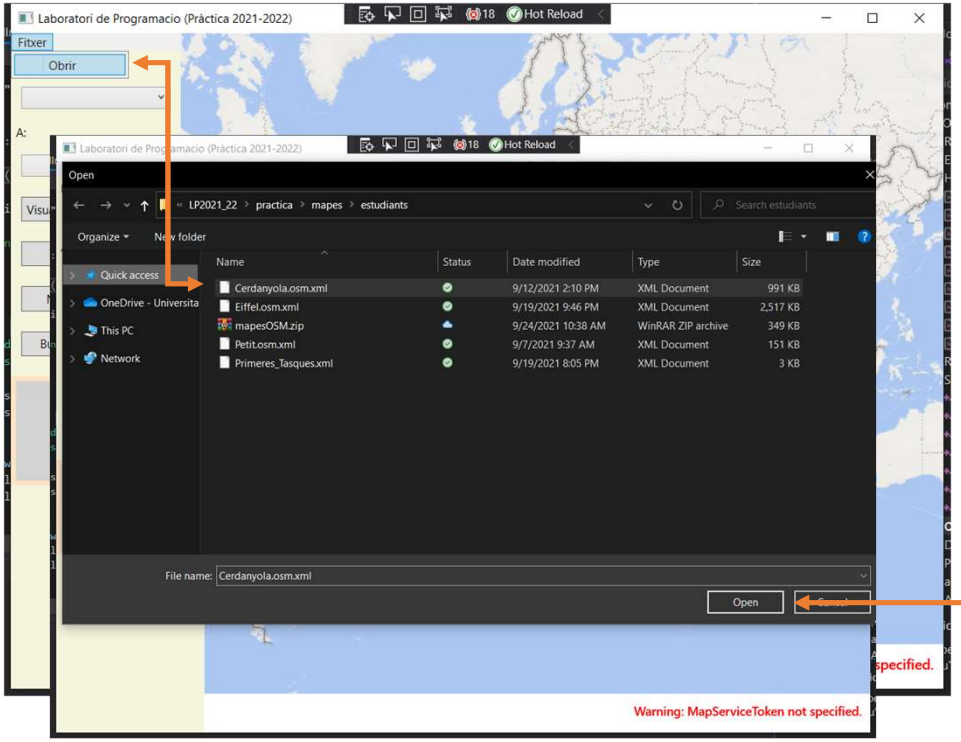
Tasques1-6 per Interfície Gràfica

1. **Un cop realitzades les Tasques descrites fins aquí, podeu començar a visualitzar punts d'interès i camins per la interfície gràfica. Però encara no heu llegit cap fitxer OSM. Si heu implementat correctament els mètodes i atributs que us hem detallat, el vostre resultat hauria de representar dos punts d'interès amb un camí.**

```
// … MapaRender.h codi
MapaRender::MapaRender() {

    this->m_xml4osm_util = new XML4OSMUtil();

    // TODO: Cal que instancieu m_mapaBase amb la vostra MapaSolucio
    this->m_mapaBase = nullptr;
}

// … MapaRender.h codi
```

Al Constructor de MapaRender, cal que instancieu m_mapaBase **amb el vostre MapaSolucio**

Agafar dades reals de OSM...




```
void parsejaXmlElements(std::vector<XmlElement>& xmlElements)
```

Llegeix les dades del vector construint les estructures internes **(Tasca 7)** per guardar: punts d'interés i camins

Informació real que prové del XML en format de vector de XmlElement. Veure Tasca 1 i Tasca 2.

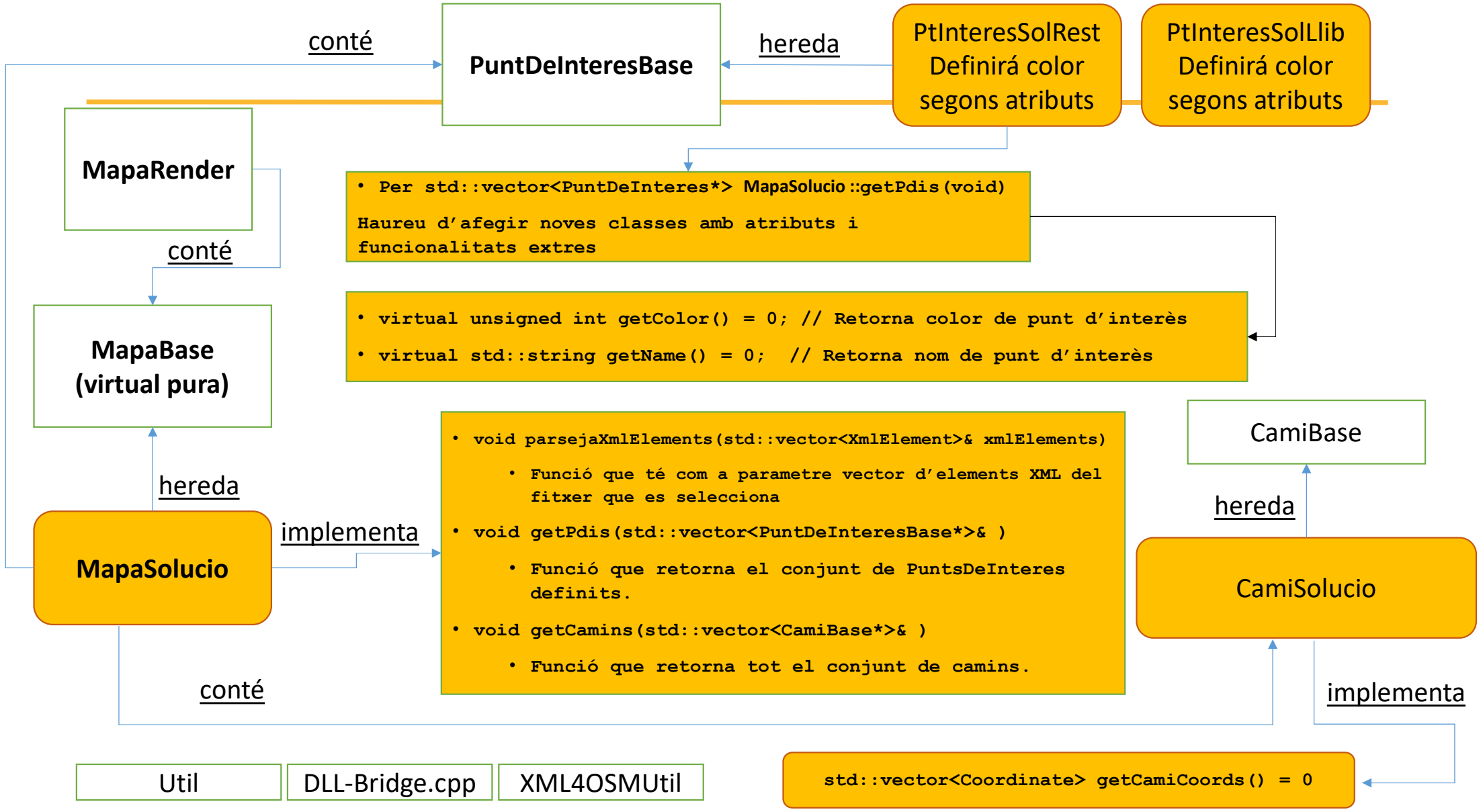
```
void getPdis(std::vector<PuntDeInteresBase*>&)
```
Tasca 8

Retorna a la interficie gràfica els punts d'interès desats a l'estructura interna

```
void getCamins(std::vector<CamiBase*>&)
```
Tasca 9

Retorna a la interficie gràfica els camins desats a l'estructura interna

- Modificar classe MapaSolucio per emmagatzemar les dades a partir d'estructures dinàmiques internes: Definició de l'estructura de dades interna necessària **Tasca 7**


conté
PuntDeInteresBase
hereda
PtInteresSolRest
Definirá color
segons atributs
PtInteresSolLlib
Definirá color
segons atributs
MapaRender
conté
• Per std::vector<PuntDeInteres*> MapaSolucio::getPdis(void)
Haureu d'afegir noves classes amb atributs i
funcionalitats extres
• virtual unsigned int getColor() = 0; // Retorna color de punt d'interès
• virtual std::string getName() = 0; // Retorna nom de punt d'interès
MapaBase
(virtual pura)
hereda
implementa
MapaSolucio
• void parsejaXmlElements(std::vector<XmlElement>& xmlElements)
• Funció que té com a parametre vector d'elements XML del
fitxer que es selecciona
• void getPdis(std::vector<PuntDeInteresBase*>&)
• Funció que retorna el conjunt de PuntsDeInteres
definits.
• void getCamins(std::vector<CamiBase*>&)
• Funció que retorna tot el conjunt de camins.
CamiBase
hereda
CamiSolucio
conté
implementa
Util
DLL-Bridge.cpp
XML4OSMUtil
std::vector<Coordinate> getCamiCoords() = 0

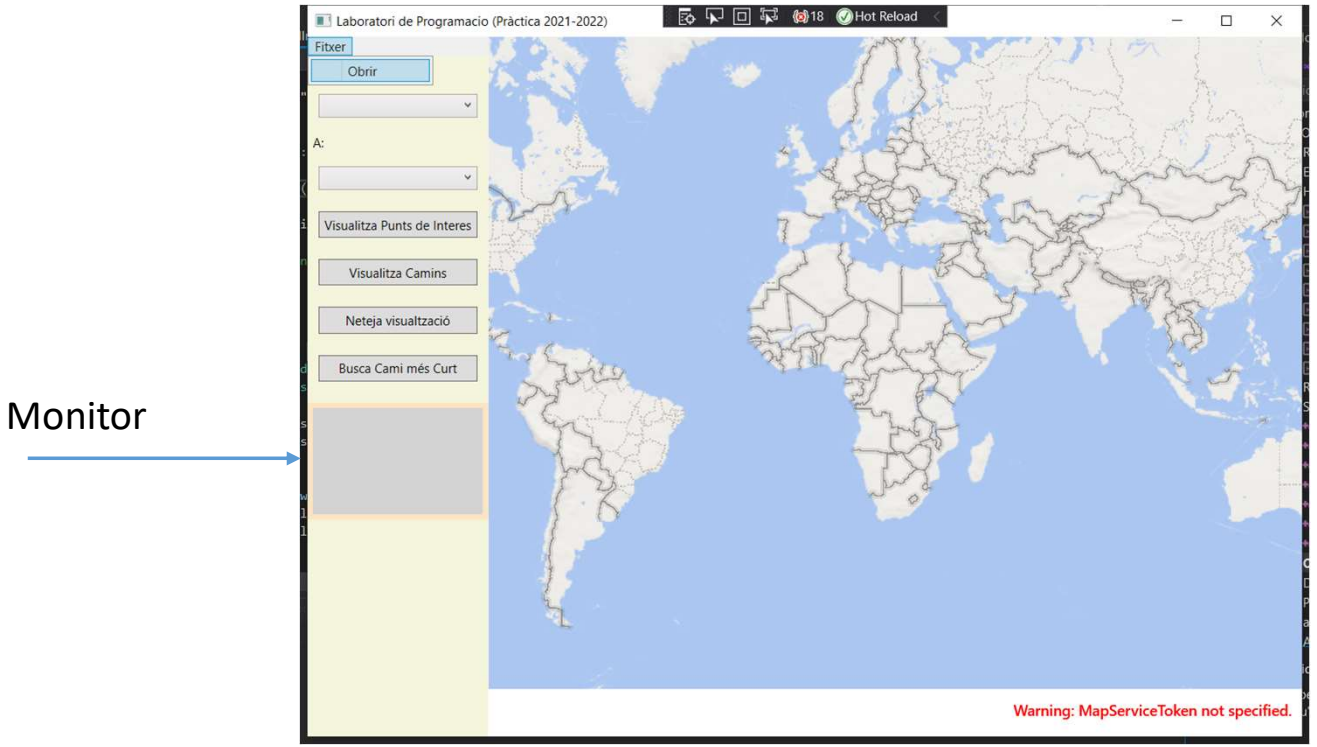
Common.h

Common.h

```
typedef std::pair<std::string, std::string> PAIR_ATTR_VALUE;
typedef std::pair<std::string, std::vector<PAIR_ATTR_VALUE>> CHILD_NODE;

typedef struct {
    double lat;
    double lon;
} Coordinate;

typedef struct {
        std::string id_element;
        std::vector<PAIR_ATTR_VALUE> atributs;
        std::vector<CHILD_NODE> fills;
} XmlElement;
```

Classe Util

1. `Util::escriuEnMonitor(std::string text)`: permet visualitzar pel monitor de la interfície el text que escriviu pel paràmetre ***text***.



Classe Util

2. `Util::kvDeTag(std::vector<PAIR_ATTR_VALUE>& atributsTag)`: us permet llegir els sub-elements de tipus **<tag>** de manera fàcil.

Aquesta funció s'encarrega d'iterar tots els atributs del tag (és a dir, `k="clau" v="valor"`), i en retorna un parell on el primer valor es "clau" i el segon "valor".

Com a paràmetre, heu de facilitar-li tots els atributs del tag que esteu computant. Torneu a fer una ullada a l'exemple donat just abans de la descripció de la Tasca 1.

Entrega a Caronte

Ara que heu implementat les vostres funcionalitats a la interfície gràfica, heu de fer l'entrega de la vostra implementació a la plataforma de Caronte.

Recordeu: L'ús de la interfície gràfica no s'utilitza per l'avaluació de la nota al projecte de l'assignatura. És més, podeu treballar directament amb els fitxers que se us proporciona a Caronte per l'avaluació automàtica. Tant mateix, sense visualitzador, segurament us serà difícil comprovar que camins i punts d'interès són inicialment els correctes, i Caronte conté els seus propis tests.

Recordeu: Tot i que aquesta presentació se us facilita les tasques que heu de fer, una descripció més detallada la podeu trobar en l'enunciat penjat a Caronte. Per qualsevol dubte, recordeu que a més de l'ajuda del professorat, teniu un fòrum especial pel projecte a Caronte per el seu bon ús.