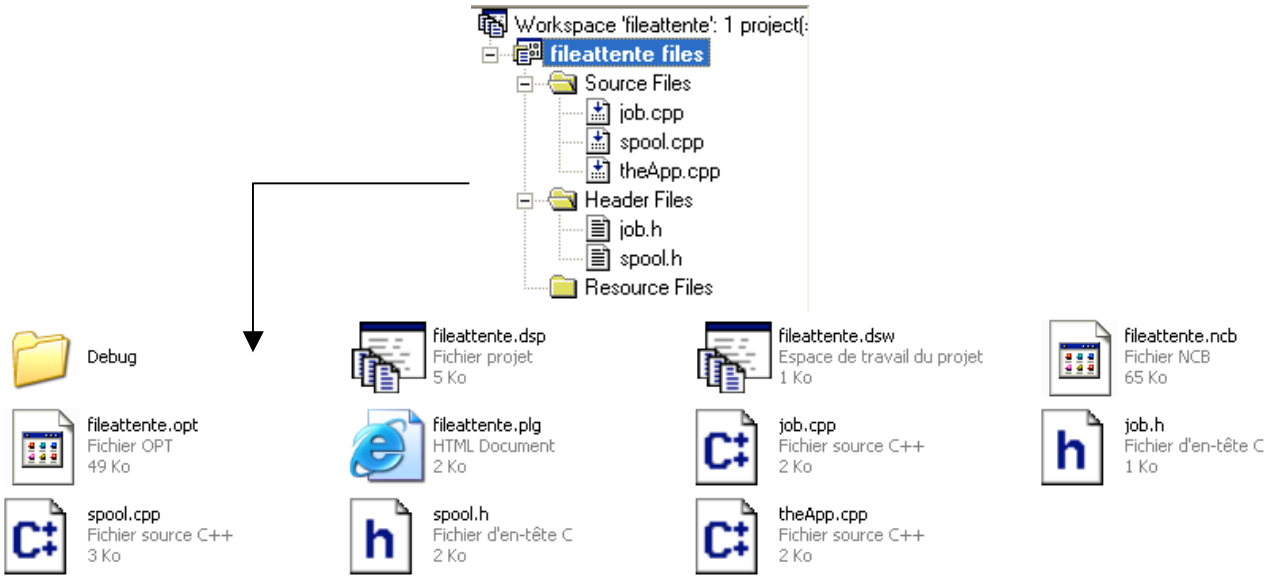
TP2
SLAM2
File d'attente

Dans le cadre d'une application dédiée à un de ses clients, SecoLOG doit réaliser un module qui permet de gérer des travaux dans une file d'attente. Dans notre cas on qualifiera notre file d'attente de « spool ». En effet c'est le terme anglo-saxon qu'il convient d'utiliser ici. Il est possible de placer un critère d'urgence sur le travail en attente dans le « spool », de manière à ce que celui-ci soit prioritaire dans la file. Un travail prioritaire dans la file est automatiquement placé en tête de file. Si un travail prioritaire est déjà présent dans la file alors il se place juste derrière.
Les travaux non prioritaires sont ensuite placés derrière les prioritaires dans leur ordre d'arrivée.
La file d'attente utilise le principe FIFO pour tous les types de travaux. La file utilise une structure doublement chaînée (suivant/précédent) pour relier les travaux entre eux.
Cette application dispose déjà d'une partie du code mis en place par un stagiaire BTS SIO. Ce stagiaire a commencé l'écriture de l'interface des différentes classes.

Le workspace de l'application (en C++) est le suivant :



Dans notre application les travaux en attente seront nommés « job ». La file d'attente correspond au « spool ».
L'application ne permet pas pour l'instant de supprimer un travail de la liste lorsqu'il est terminé. Cette fonctionnalité ne fait pas partie de votre travail.
Afin de reprendre le travail du stagiaire, il est nécessaire de comprendre ce qu'il a fait. Des annexes sont disponibles dans les documents .PDF 2 et 3, elles contiennent des informations.

Vous avez constaté que les interfaces des classes ne comportaient pas toutes les fonctions membres.
Pour cela il va falloir ajouter dans les classes les fonctions manquantes en répondant aux questions suivantes.
Dans les questions suivantes, il est nécessaire de faire apparaître à la fois le nouveau code dans l'interface et dans l'implémentation des classes concernées. De plus vous préciserez dans quel fichier vous placez votre code.

TRAVAIL A FAIRE

1. Écrivez le code des constructeurs de chaque classe.

2. Écrivez le code des destructeurs de chaque classe pour obtenir le même affichage que dans la console présentée.

3. Écrivez la méthode addJobInSpool() qui permet d'ajouter des travaux dans le spool selon leur priorité. Dans cette méthode ce fera le chaînage des travaux de la file.

```
void spool::addJobInSpool(
{
    if (this->First == NULL)
    {
        this->First = &a;
        this->Last = &a;
        temp = &a;
    }
    else
    {
```

4. Écrivez la méthode getDataOnSpool() qui affiche les informations propres au spool. Cette méthode appelle la méthode getDataOnJob().

```
void spool::getDataOnSpool()
{
    cout<<"(i) Les donnees sur le spool
```

5. Écrivez la méthode getDataOnJob() qui affiche les informations propres à un job et qui de manière récursive donne le suivant de chaque job et le précédent par la suite. Le résultat attendu dans ce cas sera fidèle au résultat de l'exécution du programme dans la console.

```
void job::getDataOnJob(
{

    cout<<"(i) Les donnees sur le job <"<<this->iNum

     this->Follow->getDataOnJob(   ;
```

Si pour des raisons de facilité, vous présentez une solution qui est non récursive, précisez-le dans votre code.

La classe spool utilise un membre privé statique. C'est une donnée propre à la classe qui peut être consultée par la classe elle-même. Cela permet à tous moment de connaître le nombre d'occurrences d'une classe.

La fonction membre statique publique de la classe spool, permet d'afficher des informations sur la classe elle-même, sans l'intermédiaire d'une de ces occurrences. Cela signifie en clair que l'on peut connaître en permanence le nombre de spool présent dans le système. Pour cela l'appel de la méthode se fera par l'intermédiaire du nom de la classe et de l'opérateur de résolution de portée ::.

```
spool::getNumberOfSpool();
```

Cependant la gestion du nombre d'occurrence d'une classe est à la charge du programmeur. Il est donc nécessaire d'être vigilant pour l'écriture du code des constructeurs et destructeurs.

La modélisation qui a donné naissance à ce projet est un peu moyenâgeuse, il est vrai que l'on aurait pu procéder avec une collection (plus simple à gérer).