Examen Licence Informatique - L3

Programmation par objets 2 (FLIN 606)

Session 1 - 6 mai 2009 - 2h

Tous documents autorisés

1 Héritage simple

Nous vous proposons d'étudier quelques éléments d'un outil de suivi du processus de développement. Nous nous concentrerons sur la représentation des problèmes pouvant survenir pendant le cycle de vie d'un logiciel (Figure 1).

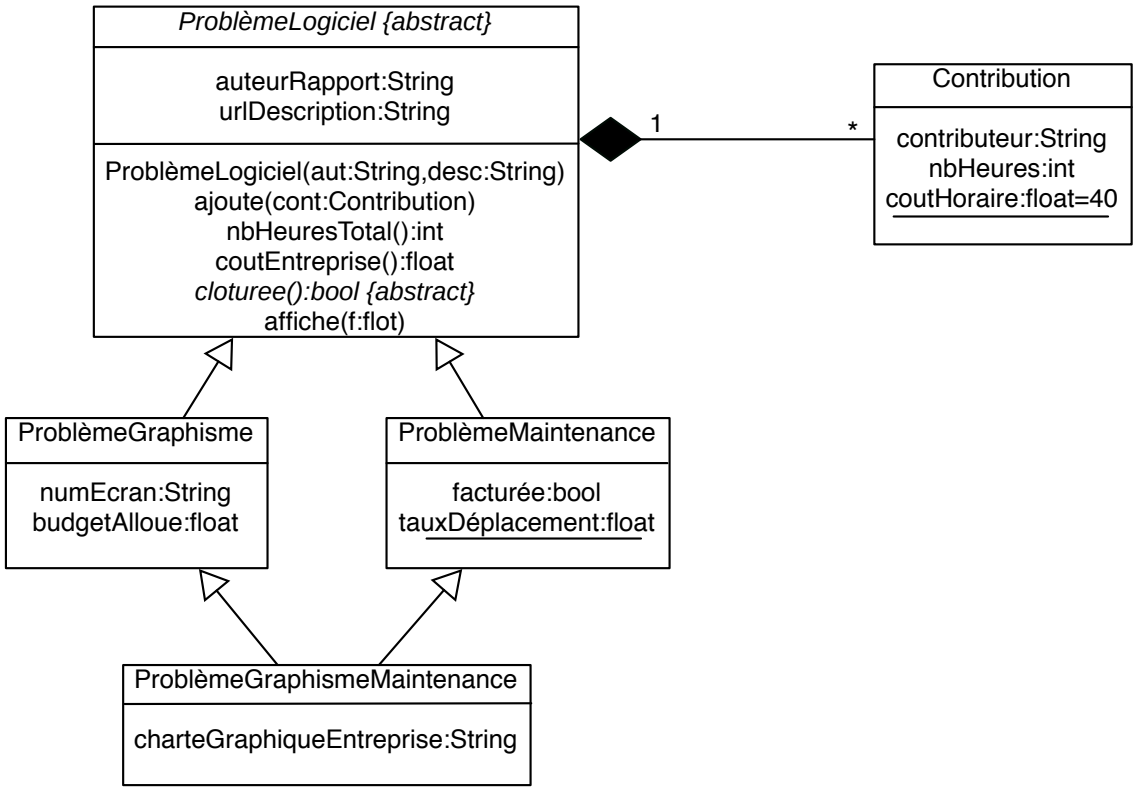

Fig. 1 – Une solution de conception pour la description de problèmes dans un logiciel. Dans les trois sous-classes les méthodes seront ajoutées au fil des questions.

Un `ProblèmeLogiciel` est décrit par le nom de l'auteur qui l'a rencontré et rapporté le premier et une url à laquelle on trouve une page qui explique le problème. Par la suite, différentes personnes peuvent apporter leur contribution à la résolution. Dans une contribution, on enregistre le nom du contributeur et le nombre d'heures passées par le contributeur. Un coût horaire fixe le coût d'une heure passée par un contributeur, il est le même pour tous dans cette modélisation simplifiée. En annexe, vous trouverez le code C++ de la classe décrivant les contributions.

Un `ProblèmeGraphisme` est un problème particulier survenant sur les aspects graphiques du logiciel. Par exemple, une police de caractères trop petite, une icône mal choisie ou d'une mauvaise taille, etc. Un tel problème est décrit plus spécifiquement par le numéro de l'écran dans lequel il apparaît et par un budget alloué à sa résolution. On considère en effet que ces problèmes sont secondaires et que le coût de leur résolution doit être limité.

Un `ProblèmeMaintenance` est un problème survenant pendant la phase de maintenance : le logiciel est déjà livré chez le client et celui-ci fait une demande d'évolution. Un déplacement des ingénieurs de développement chez le client sera nécessaire, avec un certain taux de déplacement. Lorsque le problème est résolu, on le facture au client et le booléen `facturée` devient vrai.

Un `ProblèmeGraphismeMaintenance` est un problème lié à une demande d'évolution sur

les aspects graphiques. Aux éléments précédemment décrits, on ajoute un texte décrivant la charte graphique de l'entreprise, qui devra être prise en compte pour la résolution du problème.

Question 1.1 (classes) *En respectant strictement la modélisation proposée dans la Figure 1, Ecrivez en C++ (mais n'écrivez dans cette question que ce qui vous est demandé) l'interface et le cas échéant l'implémentation pour les quatre classes des éléments suivants :*

- *l'entête des quatre classes (dont les noms, relations d'héritage, etc.),*
- *les attributs ; la liste des contributions doit impérativement être programmée avec un* `vector` *de la librairie STL,*
- *les constructeurs (justifiez leur implémentation et le choix ou non de faire un constructeur par copie) ; pour les classes* `ProblèmeGraphisme`*,* `ProblèmeMaintenance` *et* `ProblèmeGraphismeMaintenance`*, le modèle ne contient pas d'indication, vous devrez donc proposer les constructeurs les plus logiques, dont un qui soit capable d'initialiser tous les attributs de chaque classe. La liste des contributions doit être vide au moment de la création d'un problème logiciel.*
- *les destructeurs (justifiez leur implémentation).*

On supposera pour la suite que les accesseurs en lecture et en écriture aux attributs existent avec les conventions usuelles de nommage (`get`*NomAttribut et* `set`*NomAttribut).*

Question 1.2 (`ajoute`)
Ecrivez pour la classe `ProblèmeLogiciel`*, la méthode* `ajoute`*, qui ajoute une contribution au problème.*

Question 1.3 (`affiche` et surcharge d'opérateur)
Ecrivez pour les trois plus hautes classes représentant les problèmes logiciels, la méthode `affiche`*. Proposez ensuite une surcharge de l'opérateur d'insertion dans un flot pour afficher les problèmes. Quel que soit le problème, cette méthode affiche, sur un flot de sortie quelconque passé en paramètre, l'url de la description, le coût pour l'entreprise et si elle est clôturée.*

Question 1.4 (`coutEntreprise`)
Ecrivez pour les trois plus hautes classes représentant les problèmes logiciels, la méthode `coutEntreprise`*. Pour un problème logiciel quelconque, cette méthode retourne la somme des coûts des contributions. Le coût d'une contribution est le nombre d'heures multiplié par le coût horaire. Pour un problème de maintenance, on majore cette somme en appliquant le taux lié aux déplacements des ingénieurs.*

Question 1.5 (`cloturée`)
Ecrivez pour les trois plus hautes classes classes représentant les problèmes logiciels, la méthode `cloturee` *retournant vrai quand le problème est considéré comme résolu. Un problème de graphisme est cloturé automatiquement si le coût pour l'entreprise dépasse le budget alloué. Un problème de maintenance est cloturé lorsqu'il a été facturé au client.*

2 Héritage multiple

Question 2.1 (Héritage virtuel)
Indiquez ce qui assure de faire de l'héritage virtuel pour la classe `ProblemeGraphismeMaintenance`*. Précisez quelles méthodes doivent obligatoirement figurer et faites une proposition d'implémentation pour ces méthodes. Il ne doit pas y avoir de conflits au moment de la compilation de cette classe (indépendamment d'un* `main`*). Faites un schéma mémoire d'une instance de cette classe.*

Question 2.2 (Héritage répété)
Indiquez ce qu'il faut changer dans vos classes afin de faire de l'héritage répété : vous devriez indiquer des modifications dans les entêtes (y compris des deux super-classes), les constructeurs, peut-être certaines méthodes. Justifiez vos modifications. Faites un schéma mémoire d'une instance de cette classe.

Question 2.3 (Ordre d'appel des constructeurs)
Indiquez, dans les deux cas d'héritage (virtuel et répété), l'ordre d'appel des constructeurs lors de la création d'une instance de `ProblemeGraphismeMaintenance`.

3 Généricité paramétrique

Une entreprise souhaite développer un logiciel permettant l'affichage d'un tableau électronique d'annonces. Ces annonces n'ont pas un type précis, elles disposent d'une méthode `bool cloturee()`, et pourraient être des rapports de problèmes dans une équipe de développement logicielle, des offres de stages dans une université, des informations de mobilisation concernant un mouvement de grève, etc.

Question 3.1 *Proposer une classe* `TableauAffichage`*, paramétrée par le type des annonces qui peuvent y figurer. Les contraintes qui vous sont imposées pour écrire cette classe sont les suivantes :*

- *les annonces sont stockées dans un* `vector`*,*
- *une méthode* `accrocher` *permet d'ajouter une nouvelle annonce,*
- *une méthode* `décrocher` *permet de retirer une annonce rangée à une certaine position (entière) à condition qu'elle ne soit pas clôturée,*
- *une méthode* `affiche` *permet de visualiser toutes les annonces.*

Question 3.2 *Ecrivez un programme* `main` *qui crée un tableau d'affichage de* `ProblèmeLogiciel`*, lui ajoute deux problèmes, en retire un et affiche ensuite le tableau d'affichage.*

Nous vous rappelons ci-après quelques méthodes de la classe paramétrée `vector<T>` avec une signature simplifiée.

```
bool empty () const;
int size();
iterator begin ();
void push_back ( const T& x );
T operator[] ( int n );
iterator erase ( iterator position );
```

Annexe

Contribution.h

```
class Contribution
{
private:
string contributeur;
int nbHeures;
static float coutHoraire;
public:
Contribution();
Contribution(string c, int nbh);
virtual ~Contribution();
virtual string getContributeur() const;
virtual void setContributeur(string c);
virtual int getNbHeures() const;
virtual void setNbHeures(int c);
static float getCoutHoraire();
static void setCoutHoraire(float c);
};
```

Contribution.cc

```
float Contribution::coutHoraire=40;
Contribution::Contribution(){contributeur="inconnu"; nbHeures=0;}
Contribution::Contribution(string c, int nbh){contributeur=c; nbHeures=nbh;}
Contribution::~Contribution(){}
string Contribution::getContributeur()const{return contributeur;}
void Contribution::setContributeur(string c){contributeur=c;}
int Contribution::getNbHeures() const{return nbHeures;}
void Contribution::setNbHeures(int nbh){nbHeures=nbh;}
float Contribution::getCoutHoraire(){return coutHoraire;}
void Contribution::setCoutHoraire(float c){coutHoraire=c;}
```