ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Spécialité informatique
2e année

Génie Logiciel et Patrons de conception

Travaux Dirigés

Principes de conception objet	Règles de conception objet	Principes d'organisation en paquets
1. **S** ingle responsibility (responsabilité unique). 2. **O** pen-closed (ouverture / fermeture). 3. **L** iskov substitution (substitution totale). 4. **I** nterface segregation (ségrégation d'interface). 5. **D** ependency inversion (inversion de dépendance). 6. Loi de Déméter	1. Minimiser l'accessibilité aux membres des classes. 2. Encapsuler ce qui varie. 3. Programmer pour une interface, pas une implémentation. 4. Privilégier la composition à l'héritage.	1. Équivalence livraison / réutilisation 2. Réutilisation commune 3. Fermeture commune 4. Dépendances acycliques 5. Relation dépendance / stabilité 6. Stabilité des abstractions

Régis CLOUARD

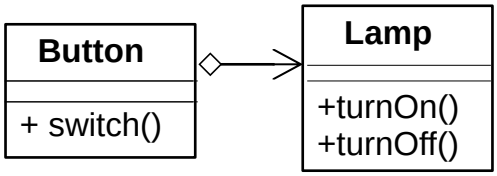
Exercice 1. Cas d'étude : un bouton allume une lampe

Considérons le cas d'un bouton et d'une lampe.

- Bouton :
  - Le bouton fonctionne sur le mode alternatif, un clic allume et le clic suivant éteint.
- Lampe :
  - À la réception du message `turnOn`, la lampe s'allume.
  - À la réception du message `turnOff`, la lampe s'éteint.

1. Conception initiale

Chaque classe est stockée dans un paquet.



Le code :

```
public final class Button {
   private boolean _isOn;
   private Lamp _lamp;
   public Button( Lamp lamp ) {
       _lamp = lamp;
       _isOn = false;
   }
   public void switch( ) {
       if (_isOne) {
           _lamp.turnOff( );
       } else {
           _lamp.turnOn( );
       }
   }
}

public final class Lamp {
   public void turnOn ( ) { ... }
   public void turnOff ( ) { ... }
}
```

2. Question

Donnez une analyse de cette conception, avantages et inconvénients ?

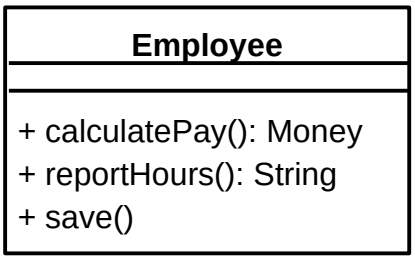
3. Modélisation

Proposez une modélisation plus SOLID.

Exercice 2. Cas d'étude : logiciel de gestion du personnel

1. Étude de cas

Soit la modélisation de la classe `Employee` avec les méthodes suivantes :



– `calculatePay()` : implémente l'algorithme pour calculer le montant du salaire en fonction des heures travaillées, du statut de l'employé, etc.

– `reportHours()` : retourne une chaîne de caractères qui sera ajoutée à un rapport que les managers utilisent pour vérifier que les employés ont travaillé le bon nombre d'heures.

– `save()` : enregistre les données de l'employé dans la base de données de l'entreprise.

2. Question

Donnez une analyse de la solution.

3. Question

Proposez une refonte (*refactoring*) de cette modélisation qui respecte les principes SOLID.

Exercice 3. Principe d'ouverture-fermeture

1. Soit la méthode suivante :

```
public double totalPrice( Parts[] parts ) {
   double total = 0.0;
   for (Part p: parts) {
       total += p.getPrice();
   }
   return total;
}
```

L'objectif de la méthode est de calculer le prix total de tous les articles présents dans le tableau article. Elle utilise la classe :

```
public class Part {
   private double _price;
   public void setPrice( double price ) {
       _price = price;
   }
   public double getPrice() {
       return _price;
   }
}
```

2. Question

Cette modélisation respecte-t-elle le principe d'ouverture fermeture ?

3. Modification : nouvelle politique de prix

Maintenant supposons que le département financier décrète que les cartes mères et mémoires sont majorées d'une taxe (respectivement 45 % et 27 %), les autres produits restant inchangés. La méthode est changée en :

```
public double totalPrice( Part[] parts ) {
   double total = 0.0;
   for (Part p: parts) {
       if (p instanceof Motherboard) {
           total += (1.45 * p.getPrice());
       } else if (p instanceof Memory) {
           total += (1.27 * p.getPrice());
       } else {
           total += p.getPrice();
       }
       return total;
   }
}
```

4. Question

Cette nouvelle modélisation respecte-t-elle le principe d'ouverture fermeture ?

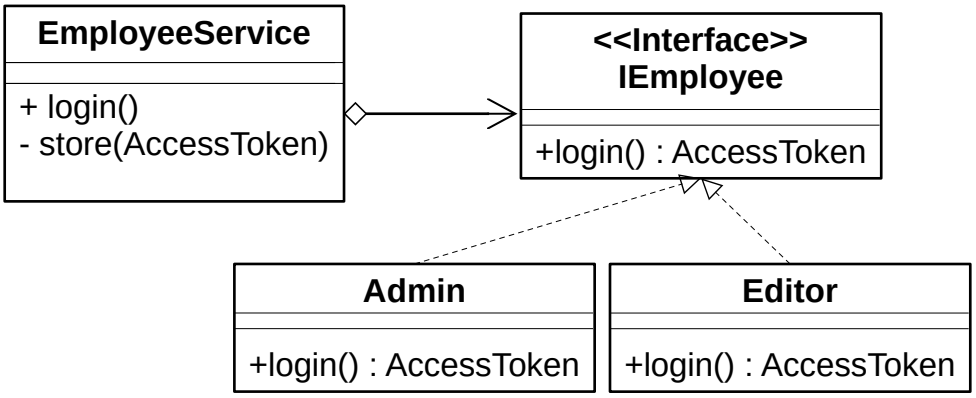
5. Modélisation

Proposez une nouvelle modélisation.

Exercice 4. Authentification d'employés

1. Étude de cas

Soit la modélisation suivante d'un service d'employés accessible à des employés devant s'authentifier par un login avant d'accéder au service :



La méthode `login()` de `EmployeeService` est :

```
public void login() {
  AccessToken token = _employee.login();
  store(token);
}
```

2. Question

On veut ajouter les employées avec les droits visiteurs qui n'ont pas besoin d'un `AccessToken`. Pour cela, on ajoute une implémentation `Guest` à l'interface `Iemployee`.

Quel est le problème de cette solution ?

Exercice 5. Logiciel de dessin vectoriel

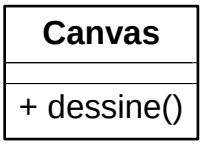
1. Étude de cas

On souhaite créer un logiciel de dessin vectoriel permettant de manipuler des :

- points,
- carrés,
- et lignes.

1.1. Hypothèse

Il existe une classe Canvas qui définit un espace de dessin :



2. Question

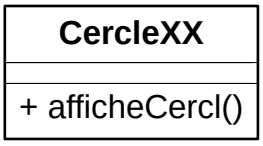
Proposez un diagramme de classes permettant de dessiner le contenu d'un espace de travail contenant les formes Point, Carré et Ligne.

3. Question

Donnez le code java des trois classes `Canvas` et `Forme` et `Point`.

4. Question

Supposons maintenant que vous deviez implémenter un nouveau type de forme tel que le cercle. Vous voulez réutiliser la classe `CercleXX` ci-dessous uniquement disponible sous la forme d'un fichier compilé `.class` donc non modifiable.



Donnez la nouvelle modélisation.

5. Question

Compléter le Java (et C++) résultant.

Exercice 6. Patron Fabrique

1. Cas d'étude : un jeu stratégie en temps réel

Dans ce type de jeu, la plupart des unités permettent de faire la guerre aux autres joueurs. Le vainqueur sera le dernier joueur disposant encore d'unités sur la carte. Généralement, les unités sont construites dans des bâtiments (usines, casernes…) grâce à des ressources (minerais, bois…) qu'il faut prélever sur la carte.

1.1. Classe abstraite Unite

Pour cela, nous définissons une classe abstraite `Unite` disposant d'un nom, d'un coût de construction, d'une précision d'attaque, d'une faculté d'esquive, etc.

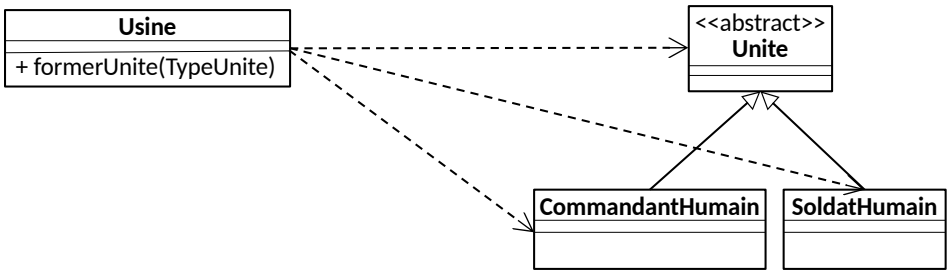
Des classes concrètes sont `Commandant` et `Soldat`.

1.2. Classe Usine

Une classe `Usine` va permettre de créer ces unités grâce à la méthode :

`formerUnite(String TypeUnite)`

La première idée qui vient à l'esprit pour implémenter cette méthode est de tester le type d'unité souhaité passé en paramètre (un identifiant de type String) via une succession de `if` ou un `switch` et de créer un objet correspondant à l'unité demandée.



2. Question : quel est le problème de cette solution ?

3. Question : Proposez une solution SOLID

4. Nouveau besoin

Notre jeu va devoir se complexifier, car les retours des bêta-testeurs indiquent que celui-ci manque de diversité. Il serait intéressant de créer plusieurs races extra-terrestres ayant chacune des unités (Soldat, Commandant…) et des bâtiments.

5. Question : proposez une modélisation SOLID

Quel patron de conception est adapté à cette modélisation ?

Exercice 7. Logiciel de dessin vectoriel

1. Étude de cas

On souhaite poursuivre le développement du logiciel de dessin vectoriel de l'exercice 5 en se basant sur 2 boîtes à outils graphiques natives qui dépendent du système sur lequel fonctionnera le logiciel : X11 pour Linux et MFC pour Windows.

1.1. Hypothèse

Chaque boîte est disponible sous la forme d'une bibliothèque compilée et d'une classe prédéfinie et non modifiable qui possède les méthodes (fictives) suivantes :

- Classe X11 : `drawRect(x1,y1,x2,y2); drawCircle(x,y,r);`
- Classe MFC : `draw_rectangle(x1,y1,x2,y2); draw_circle(x,y,r);`

2. Question 1

Proposez un diagramme de classes permettant de dessiner le contenu d'un espace de travail composé uniquement de rectangles et de cercles et qui pend en compte le système d'accueil.

3. Question 2

Les classes X11 et MFC sont très gourmandes en occupation mémoire. De plus, il n'y a aucune raison pour qu'il ait plusieurs instances de la classe X11. Quelle solution proposez-vous ?

4. Question 3

La modélisation précédente pose le problème de la destruction de l'instance de X11 ou MFC quand il n'y a plus d'instances de figure. Le problème : qui détruit l'instance de la classe X11 ou MFC ? L'avantage du ramasse-Java miette est ici incontestable. Quelle solution peut permettre de résoudre le problème en C++ ?

Exercice 8. Logiciel de calcul scientifique

1. Étude de cas : conception d'un logiciel de calcul scientifique

Il s'agit d'écrire un logiciel de calcul scientifique de style Matlab, Octave, Scilab ou R qui permet de gérer des expressions arithmétiques avec variables lues au clavier.

Par exemple, on veut pouvoir représenter des expressions littérales du genre :

```
a=(5-x)*(3+4)
```

Dans un premier temps, on se limitera aux expressions avec opérateurs binaires : +, -, *, /.

2. Question

Proposez une modélisation pour les expressions arithmétiques.

3. Question

Donnez le code Java pour construire l'expression `a=(5-x)*(3+4)`

4. Question

Donnez le code Java des classes construites.

5. Question

Proposez une modélisation pour l'évaluation des expressions (ie, calcul de la valeur d'une expression).

6. Question

Ce logiciel doit offrir la possibilité aux utilisateurs d'ajouter eux-mêmes de nouvelles opérations sur les expressions sous la forme de plugins ; par exemple, la simplification d'expression, l'affichage en préfixé ou post-fixé. Proposez une nouvelle modélisation.

7. Question

Proposez une fonction de parcours du composite pour récupérer les opérandes une à une.

Exercice 9. Multi-dispatching

Java et C++ n'implémentent que le simple dispatching. De ce fait, le polymorphisme ne peut se faire que via le membre d'appel de la méthode. Donc, si le polymorphisme dépend de plusieurs types d'objet cela n'est pas réalisable dans ces langages.

D'autres langages tels que le Clisp permettent le multi-dispatching. Ils prennent les membres d'appel en paramètres de la méthode et autorisent plusieurs types de membres à l'appel de la méthode. En Java, il faut donc trouver une autre solution pour cela.

1. Étude de cas

Jeu Chifoumi : papier, pierre, ciseaux

2. Question

Proposez une modélisation de ce jeu à partir du code de la méthode test unitaire suivant :

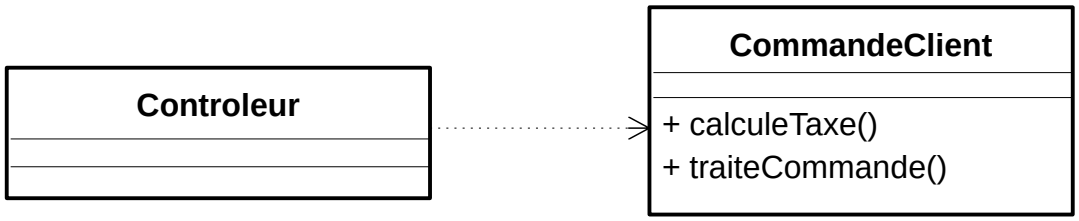
```
enum Outcome { WIN, LOSE, DRAW; }

public final class testPaperScissorsRock {
   @Test
   public void testCompeteWithAPredefinedSequence() {
       static final int SIZE = 20;
       static final int ORACLE = 11;
       int i = 0;
       int score = 0;
       ItemGenerator.setRandomSeed(0);
       for(; i < SIZE; i++) {
           Item a = ItemGenerator.newItem();
           Item b = ItemGenerator.newItem();
           score += a.compete(b);
       }
       assertEquals(ORACLE, score);
   }
}

class ItemGenerator {
   private static Random _rand = new Random();

   pubic static setRandomSeed( int seed ) { _rand.setSeed(seed); }

   public static Item newItem() {
       switch(_rand.nextInt(3)) {
       default:
       case 0: return new Scissors();
       case 1: return new Paper();
       case 2: return new Rock();
       }
   }
}
```

Exercice 10. Logiciel de commerce électronique

1. Étude de cas

On dispose d'un système de commerce électronique destiné à la France.

La modélisation courante est :



La classe Contrôleur de commandes possède la fonctionnalité suivante :

- gérer les commandes : elle identifie l'arrivée d'une commande et la transmet à l'objet `CommandeClient` qui effectue le traitement.

La classe `CommandeClient` possède les fonctions suivantes :

- effectuer le calcul des taxes,
- traiter les commandes,
- et imprimer le reçu.

2. Question

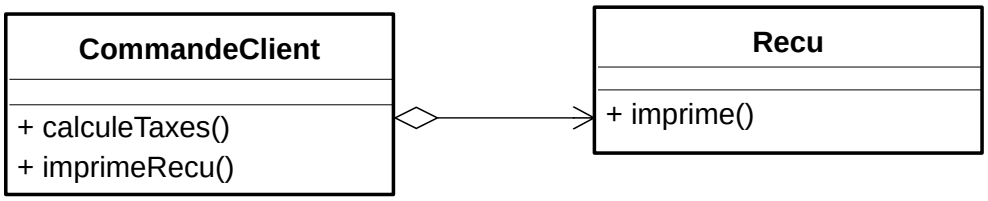
On souhaite adapter le logiciel pour la Belgique, et donc ajouter une nouvelle méthode de calcul des taxes (et uniquement la méthode de calcul des taxes).

Proposez une refonte de la modélisation pour intégrer cette nouvelle fonctionnalité.

Exercice 11. Logiciel de commerce électronique

1. Étude de cas

On continue l'étude de cas du système de commerce électronique de l'exercice 8. Nous ajoutons maintenant la possibilité d'imprimer des reçus. Pour respecter le principe de responsabilité unique, une classe `Recu` gère cette fonctionnalité.



2. Question

Nouvelle spécification : on souhaite pouvoir imprimer des reçus composés de plusieurs éléments optionnels (logo, adresse de la société, l'adresse d'un site WEB…). L'interface graphique présente un formulaire où l'utilisateur peut choisir les éléments à ajouter sur le reçu en cochant ou décochant ces options.

Quelle modélisation proposez-vous ?

Exercice 12. Logiciel de commerce électronique

1. Étude de cas

On reprend l'étude de cas du système de commerce électronique de l'exercice 8.

2. Question

À chaque insertion d'un nouveau client dans le système, on souhaite maintenant :

1) Envoyer un email de bienvenue.

2) Vérification de l'adresse du client auprès du site de La Poste.

Quelle modélisation proposez-vous ?

3. Question

Toujours au moment de l'ajout d'un nouveau client, on veut maintenant ajouter deux nouveaux services :

1) Ajout du client dans la base de données.

2) Imprimer une fiche client.

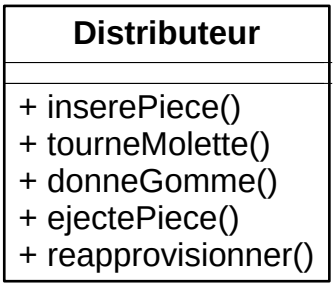
Quelle réingénierie de la modélisation proposez-vous pour que l'on puisse facilement ajouter de nouveaux services à l'avenir ?

Exercice 13. Distributeur de boules de gomme

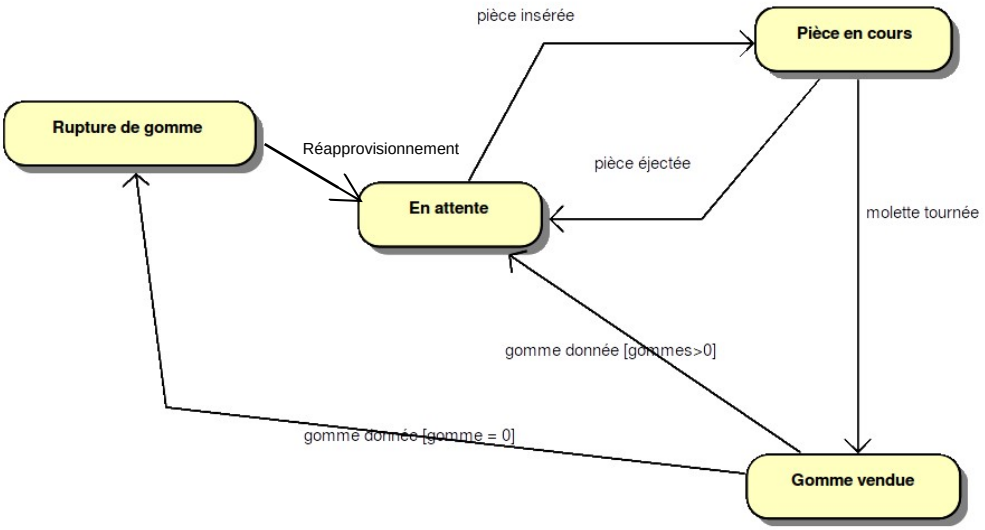
1. Étude de cas

Le logiciel à développer doit être embarqué dans un distributeur de boules de gomme disposant entre autres d'un monnayeur, d'une molette de distribution et d'un écran d'affichage.

Soit une classe Distributeur :



Voici le diagramme états-transitions d'un système de distributeur de boules de gomme. Pour l'exercice, on ne s'intéresse qu'à ce qui est affiché à l'écran du distributeur.



2. Question 1

Donnez le code Java des méthodes `inserePiece()` et `tourneMolette()`.

3. Question 2

Proposez une modélisation du distributeur qui soit SOLID.

4. Question 3

Ajouter le cas d'un mode gagnant qui distribue 2 boules au lieu d'une.

Exercice 14. Cycle chimique de l'eau

1. Étude de cas

On souhaite faire un logiciel qui modélise les propriétés chimiques de l'eau en fonction de la température.

Dans les conditions de température et de pression qui règnent dans notre système terre-atmosphère, l'eau peut exister sous trois phases différentes : liquide, solide et gazeux.

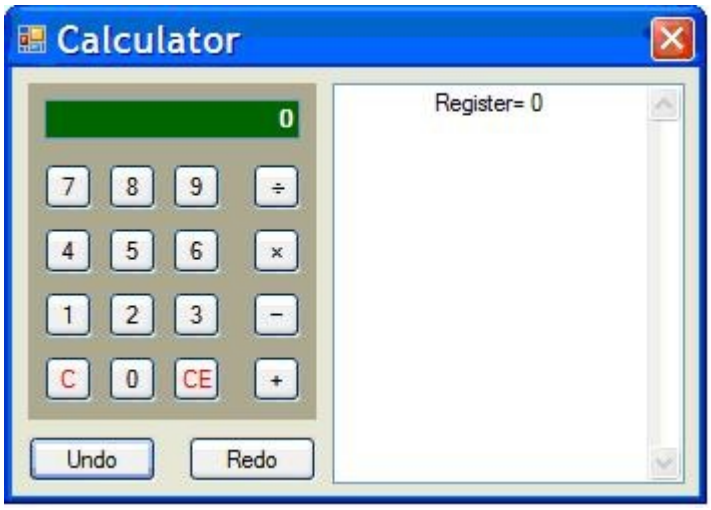
2. Question

Donnez une modélisation adaptée à la représentation du cycle chimique de l'eau en fonction de l'évolution de la température obtenue à partir d'une jauge.

Exercice 15. Calculatrice graphique

1. Étude cas

Nous souhaitons développer une calculatrice graphique présentant l'interface ci-dessous.



La calculatrice fonctionne en notation polonaise inverse (NPI) avec les quatre opérations de base. Les calculatrices NPI n'ont pas de bouton '=' et ni de parenthèses. Pour effectuer un calcul, le premier nombre est ajouté au registre de la calculatrice à l'aide du bouton '+'. Ensuite, le deuxième nombre est entré puis l'opération souhaitée en cliquant sur le bouton approprié.

Par exemple, pour multiplier 2 x 13, l'utilisateur clique sur les boutons suivants :

- Cliquez sur '2', puis sur '+' : le nombre '2' apparaît à l'écran.
- Cliquez sur '1', puis sur '3', puis sur '*' : le numéro '26' apparaît à l'écran.

Les calculs peuvent être chaînés en entrant le nombre suivant, puis en cliquant sur le bouton de l'opération souhaitée. Par exemple, pour soustraire 2 au résultat ci-dessus, cliquez sur '2', puis sur '-'. Le numéro '24' apparaîtra à l'écran.

La calculatrice comporte également les boutons « Undo » et « Redo ». Chaque fois que vous cliquez sur le bouton « Undo », une opération est annulée, en commençant par la plus récente. Le bouton « Redo » rétablit les opérations annulées.

2. Question1

Quelle architecture proposée vous ? Faites le diagramme de paquets avec les composants de cette architecture.

3. Question2

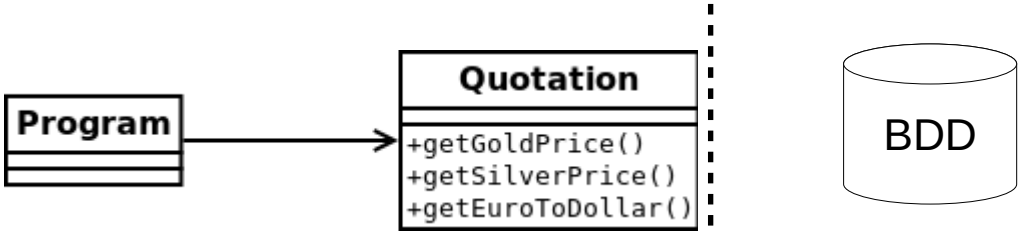
Proposez une modélisation complète de la calculatrice intégrant le mécanisme de réversion.

4. Question 3

Donnez-en le code Java.

Exercice 16. Étude de cas : client – serveur

Une application affiche le prix de produits de référence tel que l'or, l'argent et la valeur d'échange entre l'euro et le dollar qu'elle doit récupérer sur un serveur distant. Une classe `Quotation` détient les services permettant de récupérer les valeurs des produits. La cotation des produits est supposée fixe sur une journée. L'organisation initiale est la suivante :



Le serveur est ici une autre application de base de données qui tourne sur une machine du même réseau. La communication entre le client et le serveur est faite avec un protocole REST.

1. Question

Quelle modélisation proposez-vous pour mettre en place la communication entre les deux services ?

2. Question

Proposez une modélisation pour répartir le patron d'architecture MVP sur le réseau en mode client serveur.

Exercice 17. Étude de cas : logiciel de traitement de texte

Le projet concerne le développement d'un éditeur de documents texte de type WYSIWYG (*What You See Is What You Get*) tel que LibreOffice Writer ou Word. Nous allons examiner 11 situations de la conception de l'éditeur. Pour chacun vous devez donner le patron qui vous semble le mieux adapté et proposer une modélisation correspondante.

1. Architecture

L'architecture impacte fortement l'organisation du logiciel. Quel patron d'architecture vous semble le plus adapté à l'organisation d'un tel logiciel ?

2. Structure du document

Le document est vu selon un arrangement de structures graphiques nommées *Glyphes* : ligne, caractère, graphique et tableau, et non selon un arrangement plus « naturel » basé sur une organisation logique en paragraphe, section et chapitre. Cette organisation est, en effet, préférable pour des questions d'efficacité de manipulation du document.

Proposez un patron de conception adapté à la modélisation de la structure du document selon l'organisation proposée.

3. Formatage

Le formatage d'un document diffère de sa structure. Il consiste à construire la structure physique du document. Cela inclut en particulier de découper le texte du document en lignes ou de gérer les césures. Il est réalisé par une méthode de construction de la représentation logique d'un document. Néanmoins, il existe plusieurs façons de réaliser ce formatage, par exemple aligné à gauche, aligné à droite ou justifié.

Quel patron de conception permet d'implémenter au mieux l'ensemble des variantes de l'algorithme de formatage ?

4. Configuration de la fenêtre d'édition

Dans la fenêtre d'édition du document, il est possible d'ajouter une règle, une barre de statut, une jauge pour le réglage du zoom, une grille en arrière-plan, une barre d'outils, etc.

Quel patron de conception permet d'ajouter des éléments à souhait sans complexifier la fonction d'affichage de la fenêtre d'édition ?

5. Multiples « Look and Feel »

Le « look and feel » se rapporte à des standards graphiques spécifiques. Ces standards définissent la façon dont les éléments de l'interface (p. ex. scrollbar, fenêtre, jauge) se visualisent. On peut citer les L&F Nimbus et Motif pour JavaFX. Ils sont disponibles sous la forme de bibliothèque de classes prédéfinies. Le L&F est choisi au lancement du logiciel, mais peut aussi être changé à chaud. Quand le développeur veut afficher une scrollbar, l'élément concret est celui qui correspond au L&F courant, par exemple MofifScrollBar pour Motif ouNimbusScrollBar pour Nimbus.

Il s'agit de permettre au développeur de ne pas se soucier de quelle classe concrète utiliser pour créer un élément graphique en fonction du L&F courant. Quel patron permet de créer un élément graphique selon le L&F choisi sans préciser sa classe concrète ?

6. Opérations de réversion

L'une des fonctionnalités incontournable d'un traitement de texte est la réversion (annuler/refaire une action). L'utilisateur doit pouvoir entrer, supprimer, modifier du texte, des graphiques et annuler ses opérations. Quel patron permet de gérer le mécanisme de réversion ?

7. Vérification de l'orthographe

Il s'agit de lancer la vérification d'orthographe sur chacun des Glyphes. Seul le Glyphe connaît sa structuration, par exemple la liste des caractères pour une ligne de texte, ou la liste des lignes de texte pour un tableau. Il n'est pas question de modifier l'organisation en tableau, en ligne ou en mots pour intégrer cette fonctionnalité.

Quel patron permet d'accéder à la liste des mots du document sans modifier la structure choisie pour représenter le document ?

8. Extensibilité : vérification grammaticale, césures...

Il serait intéressant de pouvoir ajouter de nouvelles fonctionnalités fonctionnant comme le vérificateur orthographique : compter les mots, césure, vérificateur grammatical. L'extension se fera sous la forme de greffon (plugin) qui peuvent accéder à la structure interne d'un document. Quel patron faut-il adopter pour permettre l'extension facile de nouvelles fonctionnalités qui nécessitent le parcours des mots de la structure ?

9. Affichage des images

Les documents incluent des images qui peuvent occuper beaucoup d'espace mémoire. Quel patron pouvez-vous utiliser pour éviter les problèmes d'encombrement mémoire ?

10. Gestion d'un environnement multi-fenêtres

Il s'agit d'organiser les fenêtres en cascade ou en mosaïque. Quand on ferme une fenêtre, il faut rendre le focus sur la fenêtre directement en dessous voire réorganiser la mosaïque de fenêtres (ou enlever la fenêtre et mettre une marque en regard de la nouvelle fenêtre en cours dans le menu des fenêtres). Quel patron utiliser ici ?

11. Implémentation de l'aide sur un composant

Quel patron de conception doit être utilisé pour implémenter l'aide contextuel qui peut être obtenue en pointant la souris sur un élément de l'interface et en appuyant sur F1 ?

Exercice 18. Quiz

1. Questions

La règle n°2 de la conception est d'encapsuler les variations. **Donnez** le nom du patron de conception qui permet de répondre aux problèmes de modélisation lorsqu'il existe une variation de type :

1) Le comportement d'un service en fonction des propriétés d'un objet.

2) La nature des services rendus par un objet en fonction de la valeur d'un attribut.

3) Le parcours d'un agrégat.

4) Les parties d'un composant.

5) La liste des objets dépendant des modifications d'un objet de référence.

6) Les parties d'un service.

7) Les variantes d'un même service.

8) La liste des services d'une classe.

9) L'objet capable de rendre un service.

10) Le type concret d'un objet.

Exercice 19. Voyage inorganisé

Une touriste veut visiter des sites parmi une collection proposée par une agence de voyage. Elle souhaite voir le détail de chaque site les uns après les autres. Toutefois, elle veut pouvoir accéder à la liste dans des ordres différents : par exemple, dans un ordre du plus chemin, celui avec uniquement les églises, par ordre alphabétique, etc.

1. Question

Proposez une conception idoine.

Exercice 20. Logiciel de CAO

1. Étude de cas : logiciel d'impression de plans de CAO

Supposons que vous soyez responsable de la conception d'un système informatisé d'affichage et d'impression de plans CAO.

Le type de résolution à utiliser dépend de la puissance de l'ordinateur sur lequel est exécuté le logiciel, et plus particulièrement de la vitesse de l'unité centrale et de l'espace mémoire disponible. Le système conçu devra prendre en compte la sollicitation de l'ordinateur. Ainsi, un ordinateur puissant pourra utiliser des pilotes de grande capacité et un ordinateur moins puissant ne pourra utiliser que des périphériques de faible capacité.

Action du pilote	Faible capacité	Grande capacité
Affichage	Pilote d'affichage basse résolution (LRDD)	Pilote d'affichage haute résolution (HRDD)
Impression	Pilote d'impression basse résolution (LRPD)	Pilote d'affichage haute résolution (LRPD)

La difficulté réside dans le fait que le système doit contrôler les pilotes utilisés à partir d'une valeur de résolution donnée pour l'écran et l'imprimante.

2. Question

Proposez une solution de conception pour implémenter les deux services `paint()` et `print()`.