Grenoble INP – Ensimag
Deuxième année
2022–2023

Analyse et Conception Objet de Logiciels

Minuterie

On s'intéresse à un système « Minuterie », qui permet de déclencher une alarme après une période de temps spécifiée.

1. Une minuterie peut être pilotée par une interface graphique (cf. figure 1) qui contient :
   — trois compteurs permettant d'afficher les heures, les minutes et les secondes ;
   — quatre boutons : Mode, Incr, Start/Stop, Fermer ;
   — un indicateur d'alarme.

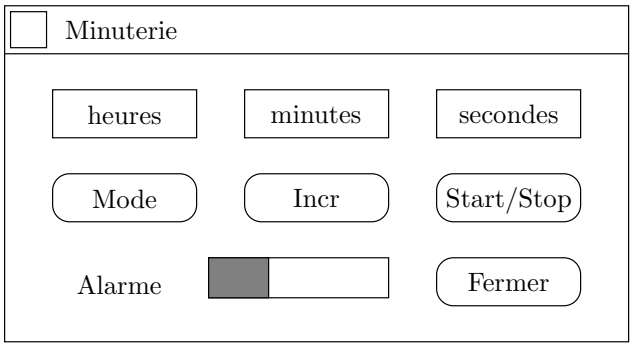


FIGURE 1 – Interface de la minuterie

2. Lorsque la minuterie est arrêtée, le bouton Mode permet de passer du mode « normal » au mode « édition heures », puis au mode « édition minutes » et enfin de revenir au mode « normal ».
3. Lorsque la minuterie est dans le mode « édition heures », le bouton Incr permet d'incrémenter ce compteur.
4. Lorsque la minuterie est dans le mode « édition minutes », le bouton Incr permet d'incrémenter ce compteur.
5. Lorsque la minuterie est arrêtée, le bouton Start/Stop permet de la démarrer.

6. Lorsque la minuterie est en marche, le bouton Start/Stop permet de l'arrêter. On se retrouve alors dans le mode « normal ». Lorsque la minuterie est en marche, elle décrémente ses compteurs d'une seconde à chaque seconde. Lorsque la minuterie atteint zéro, l'alarme est activée.
7. L'alarme, lorsqu'elle est active, peut être au niveau 1, 2 ou 3. Une fois activée, l'alarme change de niveau chaque minute, puis est désactivée par le système.
8. Lorsque l'alarme est active, le bouton Start/Stop permet de la désactiver. Après désactivation de l'alarme par l'utilisateur ou par le système, la minuterie est dans le mode « normal ».
9. À tout moment, on peut fermer la fenêtre avec le bouton Fermer.

Outil de traitement de courriers électroniques

On s'intéresse à un système de traitement de courriers électroniques. Le système permet de recevoir, envoyer, consulter, classer et stocker des courriers électroniques.

Le système permet de définir différents comptes utilisateur, qui permettent à un utilisateur d'avoir plusieurs adresses électroniques. Un compte utilisateur comporte une adresse électronique, le nom de l'utilisateur, un serveur entrant (pour recevoir les courriers) et un serveur sortant (pour envoyer des courriers). Si le système possède plusieurs comptes, il existe un compte par défaut. On peut ajouter ou supprimer des comptes.

Un courrier comporte l'adresse électronique de l'expéditeur, le nom de l'expéditeur, une date d'expédition, une liste de destinataires, un sujet, un corps et un ensemble de pièces attachées. À un courrier peut être associé une étiquette : important, personnel, à faire ou indésirable.

Pour classer ses courriers, l'utilisateur peut définir différentes boîtes à lettres. Une boîte peut contenir des courriers ou d'autres boîtes. Une boîte comporte un nom. Plusieurs boîtes sont prédéfinies : une boîte *Courriers reçus*, une boîte *Courriers envoyés*, et une boîte *Poubelle*. Il existe également une boîte *Racine* à la racine de cette hiérarchie de boîtes, qui contient toutes les autres boîtes.

L'utilisateur peut écrire un nouveau courrier, l'envoyer, lire un courrier ou marquer un courrier avec une étiquette.

L'utilisateur utilise le système par l'intermédiaire d'une interface graphique qui comporte quatre types de fenêtres (on peut avoir plusieurs fenêtres de même type) :

- une fenêtre principale permet de voir la hiérarchie des boîtes, de sélectionner une boîte, et de voir un ensemble de lignes qui identifient l'ensemble des courriers contenus dans la boîte sélectionnée. Une ligne contient le sujet, les destinataires et la date d'un courrier. Un clic sur une ligne sélectionne le courrier correspondant. À partir d'une fenêtre principale, l'utilisateur peut :
  - gérer ses comptes, lire un courrier sélectionné ou composer un nouveau courrier. La fenêtre correspondante s'ouvre.
  - vider la poubelle, imprimer, sauver ou supprimer un courrier sélectionné.
- la fenêtre de composition de courrier est ouverte lorsque l'utilisateur souhaite écrire un nouveau courrier. L'utilisateur peut attacher un ou plusieurs fichiers à ce courrier, l'imprimer et l'envoyer.
- la fenêtre de lecture de courrier est ouverte lorsque l'utilisateur souhaite lire un courrier sélectionné. Il peut alors lire ce courrier, répondre à l'expéditeur, répondre à l'expéditeur et à tous les destinataires, transférer le courrier à d'autres destinataires, l'imprimer, le supprimer ou le marquer avec une étiquette.
- la fenêtre de gestion des comptes est ouverte lorsque l'utilisateur souhaite modifier un compte, ajouter ou supprimer un compte, ou changer le compte par

défaut.
Lorsque l'utilisateur reçoit un nouveau courrier, les fenêtres principales qui affichent le contenu de la boîte *Courriers reçus* sont mises à jour.

À partir d'une fenêtre principale qui affiche les courriers d'une boîte, l'utilisateur peut sélectionner un des courriers et le déplacer vers une autre boîte (boîte destination). Si d'autres fenêtres principales affichent les courriers de cette boîte destination, ces fenêtres sont mises à jour.

L'utilisateur peut chercher un ensemble de courriers contenant certains mots clés. L'utilisateur entre ces mots clés, sélectionne la boîte dans laquelle les courriers doivent être recherchés. Si cette boîte contient d'autres boîtes, les courriers sont également recherchés récursivement dans les sous-boîtes. Le système affiche alors une ligne par courrier trouvé.

Outil d'organisation de conférences

On s'intéresse à un système logiciel qui permet d'organiser des conférences.

Introduction

Une partie du travail des chercheurs consiste à écrire des articles qui présentent le résultat de leurs recherches. Ces articles sont destinés à être publiés et présentés à leurs confrères lors de conférences.

Une conférence porte sur un certain nombre de sujets de recherche, appelés thèmes. Elle a un comité de programme, composé d'un ensemble de chercheurs, qui sélectionne les meilleurs articles pour les différents thèmes de la conférence.

Lors de la conférence, chaque article accepté pour publication est présenté par l'un des auteurs. L'ensemble des articles est publié dans les actes de la conférence.

L'objectif est d'analyser et concevoir un outil qui permet d'aider l'organisateur de la conférence, en assistant différentes étapes de l'organisation.

Connexion à l'outil

Les chercheurs peuvent se connecter à l'outil à l'aide d'un login et d'un mot de passe. Un compte comporte également une adresse électronique valide qui permet l'envoi de différents courriers.

Paramétrage de l'outil

L'outil est paramétré par l'organisateur de la conférence. Celui-ci peut en particulier :

- décider des dates importantes pour l'organisation de la conférence ;
- nommer les membres du comité de programme ;
- définir les différents thèmes de la conférence.

Soumission des articles

Les chercheurs peuvent soumettre des articles à la conférence. Pour cela, chaque auteur doit avoir un compte sur l'application. S'il n'en a pas, il peut s'en créer un. Pour un article, un des auteurs fournit le titre, les auteurs, et un fichier pdf pour le contenu de

l'article. Il indique également un ou plusieurs thèmes de la conférence auxquels l'article est rattaché.

À la première soumission, un numéro est attribué à l'article. Lorsque l'article est soumis, chaque auteur reçoit un mail qui l'en informe. Chaque auteur peut alors télécharger l'article soumis, et éventuellement en soumettre une nouvelle version (c'est-à-dire un nouveau fichier pdf, et éventuellement un nouveau titre), qui remplace l'ancienne.

Les auteurs ont accès à l'ensemble des articles qu'ils ont soumis. S'ils sont connectés, ils peuvent voir en direct les modifications éventuelles effectuées par un autre auteur.

Évaluation des articles

Les articles sont évalués par des rapporteurs, qui mettent une note et écrivent un rapport. Suivant la conférence, un article est évalué par 1, 2 ou 3 rapporteurs. Un rapporteur est soit un membre du comité de programme, soit un chercheur désigné par un membre du comité de programme.

Chaque membre du comité de programme est associé à un ensemble de thèmes de la conférence, selon ses compétences.

L'attribution des articles à évaluer se fait de la façon suivante : les membres du comité de programme ont accès à l'ensemble des articles et ils sélectionnent :
- les articles qu'ils souhaitent évaluer ;
- les articles qu'ils refusent d'évaluer.

Un membre du comité de programme doit pouvoir évaluer un article de façon impartiale, et doit donc pouvoir refuser d'évaluer un article écrit par un membre de son équipe.

En fonction des thèmes des articles, des membres du comité, et des articles sélectionnés par les membres du comité, l'application attribue alors chaque article à 1, 2 ou 3 membres du comité. Ensuite, un membre du comité peut déléguer l'évaluation de certains des articles qui lui sont attribués à un autre rapporteur.

Chaque rapporteur doit envoyer, pour chaque article qu'il évalue, sa note et son rapport via l'application avant la date limite d'envoi des rapports.

Sélection des articles par le comité de programme

Une fois que tous les rapports sont envoyés par les rapporteurs, le comité de programme se réunit pour décider quels articles sont acceptés et quels articles sont refusés. Cette décision est prise sur la base des évaluations des rapporteurs, et il y a une discussion

entre les membres du comité pour les articles tangents.

Envoi des rapports aux auteurs

Lorsque tous les articles sont examinés, le président du comité de programme entre, pour chaque article, la décision (acceptation ou refus) dans l'application. Les auteurs des articles sont alors informés de cette décision. Ils reçoivent également les rapports, qui sont anonymes (un auteur ne doit pas savoir qui a rédigé les rapports sur son article).

Soumission de la version définitive des articles

Les auteurs dont les articles sont acceptés doivent soumettre une version finale de leur article, qui tient compte des remarques des rapporteurs.

Les membres du comité de programme vérifient cette version finale et la valident. Si elle n'est pas validée, les auteurs sont invités à en soumettre une nouvelle.

Participation à la conférence

Les chercheurs peuvent participer à la conférence, c'est-à-dire soit simplement venir écouter leur confrères, soit en plus présenter un article. Si un article est accepté, un des auteurs a l'obligation d'aller le présenter à la conférence. Les participants doivent s'inscrire à la conférence via l'application. Ils doivent en particulier payer des droits d'inscription, et s'inscrire pour les repas.

Organisation du planning de la conférence

La conférence dure quelques jours, elle est organisée en sessions où les auteurs font une présentation de leur article. Les sessions sont organisées en regroupant les articles par thème. Chaque session a un président qui fait la présentation des orateurs et qui veille au respect du planning.

Dates clés

Il y a plusieurs dates importantes lors de l'organisation d'une conférence, qui sont (dans l'ordre chronologique) les suivantes.

— Date du début d'appel à articles : les chercheurs sont informés de l'organisation de la conférence et sont invités à soumettre des articles.
— Date limite de soumission : les auteurs peuvent soumettre leurs articles jusqu'à cette date.
— Date de notification d'acceptation ou refus de l'article : les auteurs sont informés si l'article est accepté ou refusé. Ils reçoivent les rapports d'évaluations écrits par les rapporteurs.
— Date limite d'envoi de la version définitive de l'article : les chercheurs doivent envoyer la version corrigée de leur article (ils doivent en particulier tenir compte des remarques des rapporteurs).
— Date de début et de fin de la conférence.

Exercices

Exercice 1.
Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et vidéo projecteur).

Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel).

Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants).

Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants.

Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer le récapitulatif horaire pour l'ensemble de la formation.

Proposer un diagramme de cas d'utilisation pour modéliser ce cahier des charges.

Exercice 2.
Proposer un diagramme d'objets correspondant au diagramme de classes représenté figure 2.

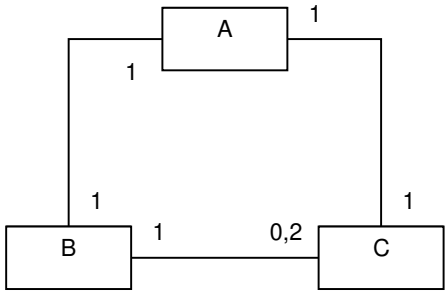


FIGURE 2 – Diagramme de classes de l'exercice 2

Exercice 3.
Proposer un diagramme d'objets correspondant au diagramme de classes représenté figure 3.

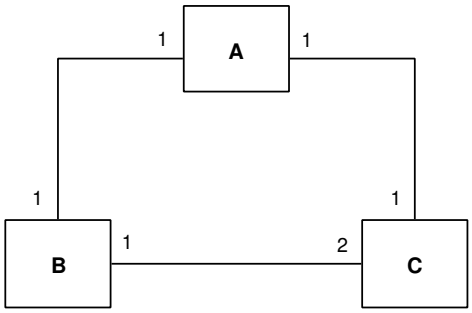
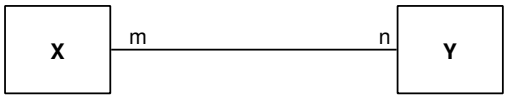


FIGURE 3 – Diagramme de classes de l'exercice 3

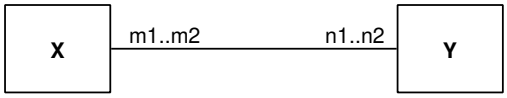
Exercice 4.
On s'intéresse aux contraintes de multiplicités associées à une relation.

1. On considère le diagramme de classes suivant.

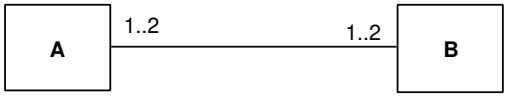


   Soit un diagramme d'objets correspondant à ce diagramme de classes. Donner une relation entre m, n, $\operatorname{Card} X$ et $\operatorname{Card} Y$.

2. Même question avec le diagramme de classes suivant.

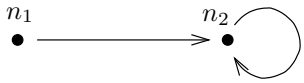


3. Soit le diagramme de classes suivant.



   Donner tous les diagrammes d'objets, utilisant des objets anonymes, tels que $\operatorname{Card} A = 3$ et $\operatorname{Card} B = 3$.

Exercice 5. Graphes
Dessiner un diagramme de classes permettant de modéliser un graphe orienté. Dessiner un diagramme d'objets correspondant au graphe suivant.

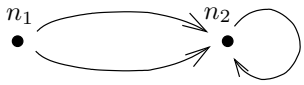


Rappel : un graphe comporte un ensemble de sommets et une relation entre ces sommets.

Exercice 6. Multi-graphes
On cherche à modéliser des multi-graphes.

Rappel : dans un multi-graphe, on peut avoir plusieurs arcs qui relient le même couple de sommets.

Dessiner un diagramme de classes permettant de modéliser un multi-graphe. Dessiner un diagramme d'objets correspondant au multi-graphe suivant.



Exercice 7. Figures géométriques

Une figure géométrique peut être un rectangle, un segment ou un cercle ou composée d'un ensemble de figures. Une figure peut être affichée, déplacée ou agrandie.

1. Dessiner un diagramme de classes qui modélise les figures géométriques.
2. Dessiner un diagramme d'objets modélisant la figure 4.
3. Dessiner un diagramme de séquence correspondant à un déplacement de la Figure 4.
4. Dessiner un diagramme de collaboration correspondant à un déplacement de la Figure 4.
5. Écrire un programme Java codant les classes et le déplacement d'une figure.

Exercice 8. Classes et interfaces Java

On cherche à modéliser la notion de classe et d'interface du langage de programmation Java. On retient les informations suivantes.

— Une classe (ou une interface) contient des attributs et des méthodes.
— Un type est soit un type de base (int, float, boolean, void), soit une classe, soit une interface.

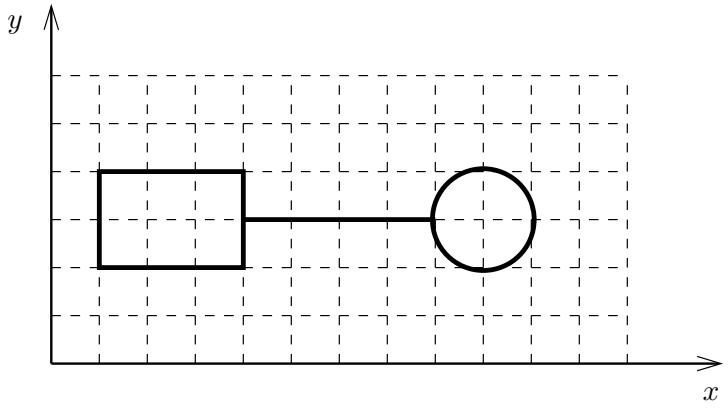


FIGURE 4 – Figure géométrique

— Un attribut a un nom et un type.
— Une méthode a un nom, un type de retour, et des paramètres.
— Un paramètre a un nom et un type.
— Une classe (ou une interface) a un nom.
— Un paquetage a un nom, et contient des classes et des interfaces.

Questions

1. Dessiner un diagramme de classes qui modélise les éléments Java définis ci-dessus.
2. Dessiner un diagramme d'objets correspondant à la classe Java suivante :

```
class Point {
   int x ; // abscisse
   int y ; // ordonnée
   boolean mêmePos(Point P){ ... }
}
```

   On cherche à afficher la classe Java `Point` sous la forme suivante :

```
class Point {
   int x ;
   int y ;
   boolean mêmePos(Point P){ }
}
```

   On suppose qu'on dispose des opérations `print(String)` et `println(String)` pour effectuer un affichage et que l'opération `+` effectue la concaténation des chaînes de caractères.
3. Dessiner un diagramme de séquence correspondant à l'affichage de la classe `Point`.

4. Dessiner un diagramme de collaboration correspondant à l'affichage de la classe `Point`.
5. Écrire un programme Java qui implémente les différentes classes ainsi que la méthode `afficher`.

Exercice 9. Clics et doubles clics d'une souris à trois boutons
On considère une souris qui comporte trois boutons : B_1, B_2 et B_3.

L'objectif de l'exercice consiste à définir un automate qui transforme les clics physiques (appuis sur les boutons B_1, B_2 ou B_3) en clics logiques. Un clic logique est de la forme `clic(b, d)` où $1 \leq b \leq 3$ indique le numéro du bouton et $1 \leq d \leq 2$ indique s'il s'agit d'un simple clic ou d'un double clic. On a un double clic lorsqu'on appuie deux fois sur le même bouton successivement et à moins de t milli-secondes d'intervalle.

1. Donner les événements produits par la séquence d'appels représentée figure 5.

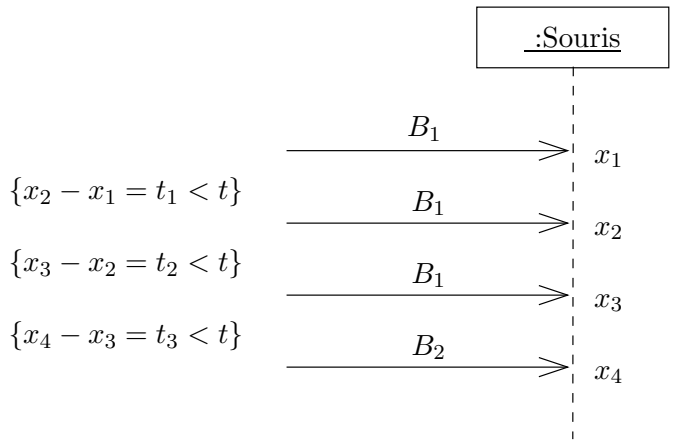


FIGURE 5 – Diagramme représentant une séquence de clics

2. Dessiner un automate qui modélise le comportement de la souris, plus précisément, qui modélise la transformation des clics physiques en clics logiques.

Exercice 10. Comportement d'une platine cassettes
On considère une platine cassettes comportant les boutons suivants : `on_off`, `play`, `stop`, `pause`, `av_r` (avance rapide), `ret_r` (retour rapide). Le fonctionnement de la platine cassettes est le suivant :

— Lorsque la platine est éteinte, `on_off` permet de l'allumer (les autres boutons sont inopérants).
— Lorsque la platine est allumée, `on_off` permet de l'éteindre ; `play` permet de la démarrer. La platine se met alors en marche, en avance normale.

— Lorsque la platine est en marche, en avance normale, le bouton `av_r` permet de passer en avance rapide ; le bouton `ret_r` permet de passer en retour rapide (dans les deux cas, on revient en avance normale avec `stop`) ; `pause` permet de faire une pause (on revient en avance normale avec `pause`).
— Lorsque la platine est en avance normale ou en pause, un appui sur `stop` permet de l'arrêter.

Dessiner un diagramme d'états-transition qui modélise le comportement de la platine.

Exercice 11. Fenêtre d'impression

Un logiciel comporte une fonctionnalité permettant d'imprimer un document. La figure 6 montre la forme de la fenêtre.

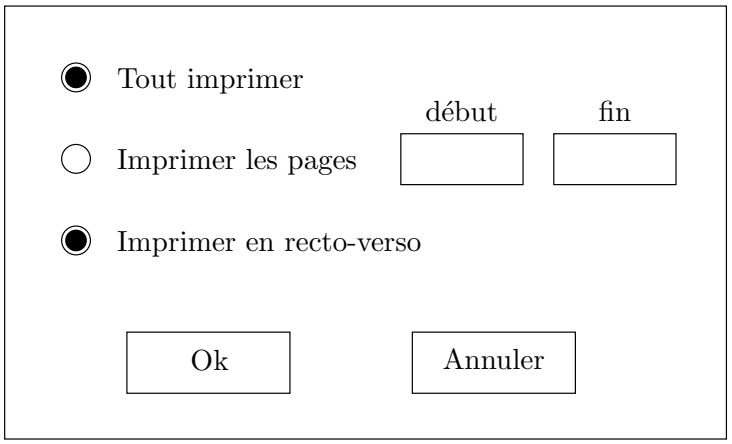


FIGURE 6 – Fenêtre d'impression

Les boutons « tout imprimer » et « imprimer les pages » ne peuvent pas être sélectionnés en même temps.

On peut indiquer la page de début et la page de fin uniquement lorsque le bouton « imprimer les pages » est sélectionné.

1. Définir les événements auxquels le système peut réagir.
2. Dessiner un diagramme d'états-transitions représentant le comportement de la fenêtre d'impression.
3. « Aplatir » ce diagramme d'états-transitions (c'est-à-dire : donner un automate équivalent qui ne comporte ni sous-état, ni sous-automates mis en parallèle).

Exercice 12. Comportement d'un bouton

On cherche à modéliser le comportement d'un bouton (classe Java `JButton`). À un bouton est associée une action, qui est exécutée lorsque l'on clique dessus. La figure 7 montre le comportement simplifié d'un bouton.

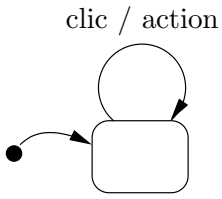


FIGURE 7 – Comportement simplifié d'un bouton

On cherche à modéliser plus finement le comportement d'un bouton, en tenant compte des éléments suivants.

- Un bouton peut être visible ou invisible. On passe d'un état à l'autre avec la méthode `setVisible(boolean)`.
- Un bouton peut être actif ou inactif. On passe d'un état à l'autre avec la méthode `setEnabled(boolean)`.
- Un clic peut être effectué uniquement si le bouton est visible et actif.
- Pour effectuer un clic, la souris doit pointer sur le bouton et on appuie sur le bouton, qui devient « armé ». L'action se déclenche lorsque l'on relâche le bouton.
- Si la souris quitte le bouton armé, celui-ci se désarme, et l'action ne se déclenche pas lorsqu'on relâche le bouton. Si la souris pointe à nouveau sur le bouton, celui-ci se réarme.

1. Décrire les événements auxquels le bouton peut réagir.
2. Dessiner un diagramme d'états-transitions qui modélise le comportement d'un bouton.
3. Que se passe-t-il lorsque l'on appuie sur le bouton, puis que l'on pointe sur le bouton avec la souris, puis que l'on relâche le bouton ?

Exercice 13. Analyse et expression des besoins de la minuterie

Le but de l'exercice est de réaliser l'analyse des besoins du système « Minuterie ».

1. Décrire les acteurs.
2. Décrire les événements auxquels le système doit réagir.
3. Identifier les passages imprécis, ambigus ou incomplets du cahier des charges, proposer une question correspondante (qui pourrait être posée aux utilisateurs) et proposer une réponse. Ce point pourra être complété au fur et à mesure qu'on répondra aux questions 4, 5 et 6.
4. Décrire les cas d'utilisation du système : dessiner un diagramme de cas d'utilisation et décrire les différents cas d'utilisation.
5. Documenter les cas d'utilisation à l'aide de diagrammes de séquence système.
6. Spécifier le comportement du système à l'aide d'un diagramme d'états-transitions.

Exercice 14. Diagramme de classes d'analyse pour la minuterie
Le but de l'exercice est de faire l'analyse de la minuterie.

1. Dessiner un diagramme de classes d'analyse pour la minuterie.
2. Dessiner un diagramme d'objets correspondant à une minuterie.

Exercice 15. Architecture pour la minuterie
Proposer une architecture pour le système « Minuterie ».

Exercice 16. Analyse et expression des besoins du système de traitement des courriers électroniques
Le but de l'exercice est de réaliser l'analyse des besoins du système de traitement des courriers électroniques.

1. Identifier les différents acteurs.
2. Identifier les cas d'utilisation du système. Dessiner un diagramme de cas d'utilisation.
3. Décrire les principaux scénarios d'utilisation du système à l'aide de diagrammes de séquence système.

Exercice 17. Diagramme de classes d'analyse pour le système de traitement des courriers électroniques
Proposer un diagramme de classes d'analyse pour le système de traitement des courriers électroniques. Préciser les attributs de chaque classe, mais pas les méthodes. Pour les associations, préciser les multiplicités, agrégations et compositions.

Exercice 18. Architecture pour le système de traitement des courriers électroniques
Proposer une décomposition architecturale pour le système.

Exercice 19. Architecture pour un système de gestion de robots
On considère une partie de la chaîne de production d'une voiture : la salle de peinture des voitures et l'entretien des robots peintres. Cette salle est constituée de robots peintres : chaque robot doit être nettoyé soit lorsque la couleur change, soit suffisamment régulièrement pour son entretien. L'application vise à assister la gestion de cette salle de peinture et la salle de nettoyage de robot en permettant de :

— attribuer un robot à la peinture d'une voiture ;
— demander le nettoyage d'un robot ;
— récupérer des informations sur les robots et les voitures.

On considère que l'application utilise/modifie :

— des données stockées dans une base de données,
— une interface graphique,
— une API qui gère le nettoyage des robots.

La base de donnée contient des informations relatives aux voitures et robots. Une voiture est définie, entre autres, par :

— un numéro d'identification,
— des dates de début et de fin de production,
— l'étape de production en cours.

Un robot est défini, entre autres, par :

— numéro d'identification ;
— son état courant : en service, au repos, hors-service, en nettoyage ;
— la date de son dernier nettoyage ;
— la couleur courante ;
— l'identification de la voiture en cours de peinture.

L'interface graphique est fixée par le client. Le visuel doit toujours apparaître sur une seule page qui contient :

— une liste de robots ;
— une liste des voitures qui sont dans la salle de peinture ;
— une fenêtre d'affichage ;
— un bouton pour lancer le nettoyage d'un robot qui doit être préalablement sélectionné dans la liste ;
— un bouton pour lancer la peinture d'une voiture par un robot qui doivent être préalablement sélectionnés dans les deux listes.

La partie « fenêtre d'affichage » permet de visualiser :

— l'acquittement ou le problème rencontré suite au lancement d'une action (peinture ou nettoyage) ;
— les informations relatives au robot ou à la voiture sélectionné par un double-clic dans une des deux listes.

1. Proposer une architecture de type MVC pour cette application.
2. Proposer une architecture en couches pour cette application.
3. Quelles sont les différences entre les deux types d'architecture logicielle ?
   Quelles sont les spécificités d'une application qui permettraient de choisir l'une ou l'autre de ces architectures ?

Exercice 20. Architecture MVC

On s'intéresse à un système qui permet d'afficher la vitesse du vent dans une fenêtre graphique. On a trois unités de vitesse possibles : *km/h*, *m/s* et *nœuds*. La fenêtre contient un champ où la vitesse est affichée, et un bouton qui permet de passer successivement d'une unité de vitesse à une autre. La vitesse est mise à jour toutes les secondes.

On a une première version de ce système, qui comporte trois classes :

— La classe `Anemometre` est la classe principale.
— La classe `UniteVitesse` est un type énuméré, qui définit trois constantes pour le choix d'unité pour la vitesse : `UniteVitesse.KMH`, `UniteVitesse.MS`, et

`UniteVitesse.NOEUD`. Cette classe contient trois méthodes :
  - `String toString()` permet de convertir une unité de vitesse en chaîne de caractères ;
  - `float coefficient()` qui fournit un coefficient pour convertir une vitesse dans une certaine unité en une vitesse en km/h ;
  - `UniteVitesse next()` qui permet de passer d'une unité de vitesse à la suivante (permutation circulaire sur les trois unités de vitesse).
- La classe `CapteurVent` permet de s'interfacer avec un dispositif physique qui mesure la vitesse du vent. Cette classe fournit une méthode `getVitesse()` qui permet de récupérer la vitesse du vent en km/h.

Le code de la classe `Anemometre` est le suivant.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Anemometre extends JFrame {

   // La vitesse mesurée courante en km/h
   private float vitesse;

   // L'unité de vitesse courante
   private UniteVitesse uniteVitesse = UniteVitesse.KMH;

   // Eléments de l'interface
   // Une étiquette pour écrire la vitesse
   private JLabel etiqVitesse = new JLabel();
   // Le bouton où est écrite l'unité choisie
   private JButton boutonUnite = new JButton(uniteVitesse.toString());

   // Permet l'interface avec le capteur de vent
   private CapteurVent capteur = new CapteurVent();

   // Un timer pour interroger le capteur de vitesse de vent à
   // intervalles spécifiés
   private Timer timer;

   /**
    * Constructeur.
    */
   public Anemometre() {
      // Mise en place de l'interface graphique
      JPanel panneau = new JPanel();
      setSize(500, 100);
      panneau.add(etiqVitesse);
      panneau.add(boutonUnite);
      this.add(panneau);
      setVisible(true);

      // Ce qui se passe lorsqu'on clique sur le bouton d'unités
      boutonUnite.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
                // On passe à l'unité suivante
                uniteVitesse = uniteVitesse.next();
                // On met à jour le texte du bouton avec le nom de l'unité
                boutonUnite.setText(uniteVitesse.toString());
                // On met à jour la vitesse
                miseAJourVitesse();
            }
        });

        // On interroge le capteur de vitesse de vent toutes les secondes
        timer = new Timer(1000, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // On récupère la vitesse depuis le capteur
                vitesse = capteur.getVitesse();
                // On met à jour la vitesse
                miseAJourVitesse();
            }
        });

        // On démarre le timer
        timer.start();
    }

    /**
     * Met à jour l'affichage de la vitesse avec la vitesse courante.
     */
    private void miseAJourVitesse() {
        float v = vitesse/uniteVitesse.coefficient();
        etiqVitesse.setText(Float.toString(v));
    }
}
```

1. Dessiner un diagramme de classes pour ce système.
2. On souhaite proposer une architecture plus modulaire pour ce système, de type MVC (Modèle – Vue – Contrôleur). On choisit une variante inspirée par certains *frameworks*, où le contrôleur est central (cf. figure 8). Ce contrôleur a accès au modèle et à la vue, mais pas l'inverse. Il n'y a aucune interaction directe entre la vue et le modèle.

   On suppose que le programme principal est de la forme suivante :

   ```
   public class AnemometreTest {
       public static void main(String[] args) {
           AnemoModele modele = new AnemoModele();
           AnemoVue vue = new AnemoVue();
           AnemoControleur ctrl =
               new AnemoControleur(modele, vue);
       }
   }
   ```

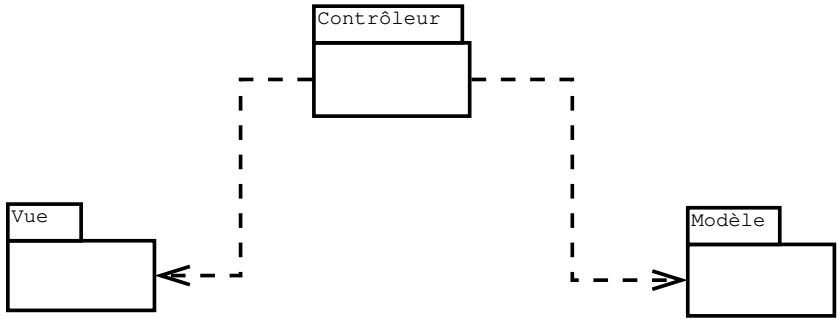


FIGURE 8 – Architecture MVC

Le but de cette question est de décomposer la classe `Anemometre` de façon à respecter l'architecture proposée.

Dessiner un diagramme de classes correspondant ainsi que le code Java des classes introduites. On utilisera les classes `UniteVitesse` et `CapteurVent` sans les modifier ni en détailler le code.

3. Proposer un programme qui permet d'afficher deux fenêtres graphiques indiquant chacune la vitesse du vent ainsi que le bouton permettant de changer l'unité de vitesse, en utilisant le même capteur. L'unité de vitesse peut être différente dans les deux fenêtres.

Exercice 21. Patron Adaptateur
Le but de l'exercice est de faire la conception du système suivant en appliquant le patron Adaptateur.

On considère un ensemble d'appareils qui peuvent être allumés ou éteints (télévision, four, radiateur... etc.). Pour chaque appareil, on a une classe Java (`Télévision`, `Four`, `Radiateur`...) qui a ses propres méthodes, différentes pour toutes les classes. Par exemple, la télévision peut être allumée avec la méthode `TVOn()`, le four peut être allumé avec `allumer()`.

On souhaite modéliser une prise multiple, avec une méthode `on()` et une méthode `off()`, qui permettent d'allumer et d'éteindre un ensemble d'appareils. On souhaite réutiliser les classes `Télévision`, `Four` et `Radiateur` *sans les modifier*.

1. Dessiner un diagramme de classes qui représente les différents éléments du système.
2. Donner le code Java des principales classes.

Exercice 22. Patron de conception Observateur
Le but de l'exercice est d'instancier le patron de conception Observateur sur des af-

fichages de trois valeurs de pourcentage (affichage textuel, par histogramme et par camemberts), cf. Figure IV. 12 page 86 du polycopié de cours.

1. Dessiner le diagramme de classes correspondant.
2. En Java, écrire les classes et interfaces `Sujet`, `Observateur`, `Pourcentage` et `Texte`.

Exercice 23. Patron de conception Interprète
On considère la grammaire abstraite suivante :

Exp	→	ExpBin | ExpUn | num | var
ExpBin	→	Exp + Exp | Exp - Exp | Exp * Exp
ExpUn	→	+ Exp | - Exp

1. Écrire un diagramme de classes correspondant à cette grammaire.
2. Écrire le diagramme d'objets correspondant à l'expression arithmétique `2 * - a`.
3. Écrire les classes Java, ainsi qu'une méthode `évaluer` permettant d'évaluer une expression arithmétique dans un environnement.
   On rappelle qu'un environnement associe à une variable sa valeur.

Exercice 24. Patron de conception Visiteur
Appliquer la patron Visiteur pour définir l'opération `évaluer` de l'exercice précédent.

Exercice 25. Patron Décorateur
On considère une hiérarchie de composants, avec une méthode `operation()`, abstraite dans la classe abstraite `Composant` (cf. figure 10).

Une méthode classique pour ajouter des responsabilités (attributs, méthodes) à des objets est l'héritage, qui consiste à définir de nouvelles sous-classes de `Composant` ou `ComposantConcret`. Avec cette méthode, ces responsabilités sont ajoutées statiquement, et à tous les objets de la classe.

On souhaite ici ajouter des responsabilités à des objets individuellement, plutôt qu'à toute une classe, et de façon dynamique. Une approche consiste à utiliser le patron *Décorateur* : on définit une sous-classe abstraite `Decorateur` de `Composant`, qui contient un `Composant`. Cette classe définit la même interface que `Composant`, soit une méthode `operation()`, qui appelle `operation()` sur le composant contenu dans le décorateur (cf. figure 11).

La classe `Decorateur` a des sous-classes `DecorateurConcret`. Un `DecorateurConcret` peut ajouter des attributs et des méthodes. La méthode `operation()` peut également être redéfinie.

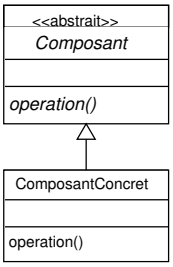


FIGURE 9 – Hiérarchie de composants

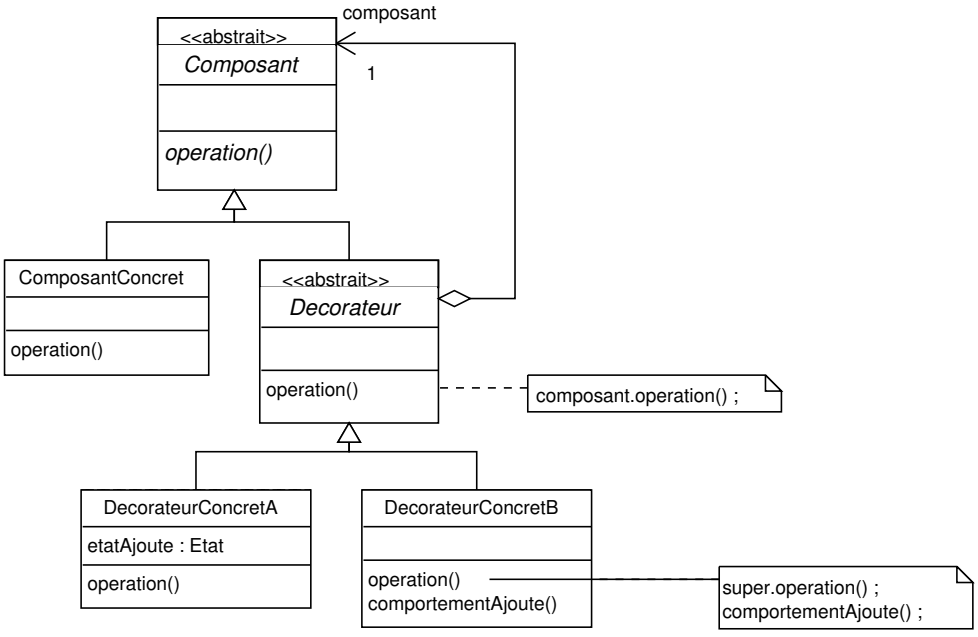


FIGURE 10 – Structure du patron Décorateur

Le diagramme d'objets représenté figure 12 montre qu'on peut appliquer plusieurs décorateurs en cascade à un objet : le composant est d'abord décoré avec `DecorateurConcretB`, puis avec `DecorateurConcretA`.

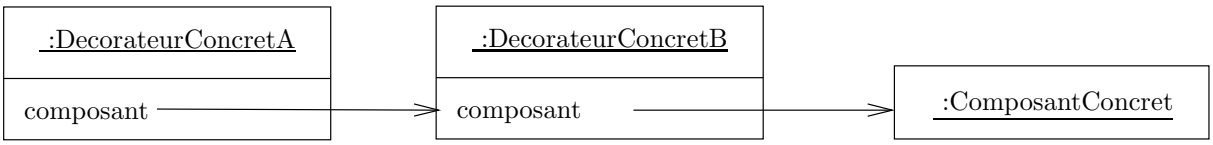


FIGURE 11 – Un diagramme d'objets

L'objectif de cet exercice est d'appliquer le patron *Décorateur* sur le problème suivant.

Un vendeur de pizzas souhaite informatiser le calcul du prix de ses pizzas. Il y a des pizzas à pâte fine et des pizzas à pâte épaisse. Outre la sauce tomate, une pizza comporte un certain nombre d'ingrédients optionnels, qui peuvent être mis en une ou plusieurs doses. Les ingrédients sont les suivants : gruyère, mozzarella, oignons, champignons, jambon, oeuf. Chaque dose d'ingrédient a un prix, de même que chaque pâte. L'objectif est de calculer le prix d'une pizza.

Pour modéliser les différentes pizzas, les deux types de pâte correspondent à deux composants. Les ingrédients sont codés à l'aide de décorateurs de pizza.

1. Proposer un diagramme de classes modélisant les pizzas qui utilise le patron *Décorateur*. Préciser les attributs et les opérations.
2. Faire un diagramme d'objets correspondant à une pizza à pâte fine avec une simple dose de jambon et mozzarella et deux oeufs (double dose d'oeuf).
3. Donner le code *Java* des principales classes du diagramme.

Exercice 26. Circuits électroniques
Dans cet exercice, on cherche à modéliser des circuits logiques. Ces circuits comportent des entrées (qui ont pour valeur soit 0 soit 1), et des portes logiques : portes *Non*, *Et*, et *Ou*. La figure 9 montre un exemple de circuit logique.

L'objectif est de modéliser et implémenter en Java de tels circuits, de sorte que lorsqu'une valeur d'entrée est modifiée, la valeur en sortie est recalculée. Si on a besoin deux fois de la valeur de sortie, sans que les entrées soient modifiées, celle-ci n'aura donc été calculée qu'une seule fois.

On souhaite pouvoir faire des tests de la forme suivante :

```
Entree e1 = new Entree(true) ;
Entree e2 = new Entree(true) ;
```

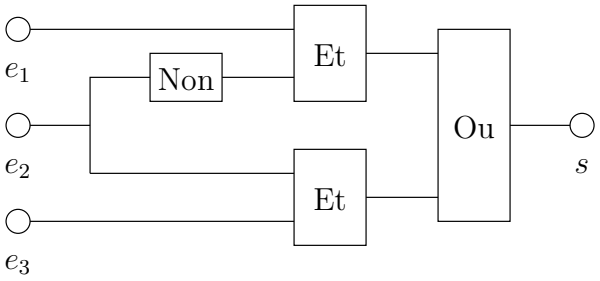


FIGURE 12 – Exemple de circuit logique

```
Entree e3 = new Entree(false) ;
Circuit c = ...
   // Construction du circuit en fonction de e1, e2, e3
System.out.println("Sortie de c : " + c.getValeur()) ;
e2.setValeur(false) ; // La sortie est recalculée
System.out.println("Sortie de c : " + c.getValeur()) ;
   // c.getValeur() ne recalcule pas la sortie
```

1. Proposer l'utilisation de deux patrons de conception pour modéliser les circuits. Dessiner un (unique) diagramme de classes correspondant l'application de ces deux patrons.
2. Donner le code Java des principales classes du diagramme.

Exercice 27. Conception de la minuterie
Le but de cet exercice est de réaliser la conception détaillée du système « Minuterie ». On décide d'utiliser différents patrons de conception : Observateur, Visiteur et État.

1. Montrer comment le patron Observateur peut être utilisé pour gérer les mises à jour de l'interface graphique.
2. Montrer comment les différents états du contrôleur peuvent être représentés à l'aide du patron État.
3. Montrer comment les actions à effectuer en réponse à des clics sur les boutons Incr, Mode et Start/Stop peuvent être représentées en utilisant le patron Visiteur.
4. Préciser le contenu des principales classes.

Exercice 28. Conception du système de traitement des courriers électroniques
L'objectif de l'exercice est de réaliser la conception du système de traitement des courriers électroniques en appliquant différents patrons de conception.

1. Proposer un patron de conception pour représenter la hiérarchie des boîtes à lettres. Dessiner le diagramme de classes correspondant.

2. Proposer un patron de conception pour gérer la mise à jour de l'affichage des boîtes lorsqu'un courrier est reçu ou lorsqu'un courrier est déplacé d'une boîte vers une autre. Dessiner le diagramme de classes correspondant.
3. On souhaite utiliser le patron *Visiteur* pour faire une recherche de courriers contenant certains mots clés. Dessiner le diagramme de classes et écrire le code *Java* correspondant. On supposera donnée dans une classe `Recherche` une méthode

```
static boolean contient
   (Courrier courrier, Set<String> liste_mots_cles);
```

qui retourne *vrai* si le courrier `courrier` contient les mots clés contenus dans l'ensemble de chaînes `liste_mots_cles`.

Exercice 29. Outil d'organisation de conférences
Le but de cet exercice est de réaliser l'analyse et la conception de l'outil d'organisation de conférences décrit page 5.

1. Analyse
   (a) Décrire les acteurs.
   (b) Déterminer les principaux cas d'utilisation du système. Dessiner un diagramme de cas d'utilisations et décrire chaque cas d'utilisation en quelques lignes claires et précises.
   (c) Décrire les principaux scénarios d'utilisation du système à l'aide de diagrammes de séquence système (faire environ quatre diagrammes).
2. Diagramme de classes d'analyse
   Proposer un diagramme de classe d'analyse. Pour chaque classe, on précisera les attributs, mais pas les opérations. Pour les associations, on précisera les multiplicités, agrégations et compositions.
3. Architecture
   Proposer une décomposition architecturale du système.
4. Diagramme d'états-transitions
   Proposer un diagramme d'états-transition qui fait apparaitre les différents états d'un article.