UNIVERSITÉ DE SHERBROOKE

DÉPARTEMENT D'INFORMATIQUE

IFT 159

Analyse et programmation

Examen final

Le lundi 15 décembre 2008
de 9h00 à 12h00

Professeur : Gabriel Girard

CAHIER QUESTIONNAIRE

- Seul un manuel de référence en C++ est permis.
- Répondez dans les espaces prévus à cet effet dans le cahier de réponses.
- La correction est, entre autres, basée sur le fait que chacune de vos réponses est :
  - claire, c'est-à-dire lisible et compréhensible pour le correcteur ;
  - précise, c'est-à-dire exacte ou sans erreur ;
  - complète, c'est-à-dire que toutes les étapes de résolution du problème sont présentes ;
  - concise, c'est-à-dire que la méthode de résolution est la plus courte possible.

1. (12 points) **Sujet : Le jeu du canon**

   Dans votre troisième travail, vous avez implanté le jeu du canon. Reprenez la spécification de ce problème pour en extraire les nouveaux types possibles. Donnez le code des nouveaux types sous forme d'enregistrements (« `struct` ») et de types énumérés (« `enum` »). Donnez aussi des exemples de code permettant d'initialiser une instance d'un des enregistrements que vous avez identifiés.

   On vous demande d'écrire un programme qui implante le jeu du canonnier. Le but de ce jeu est de tirer, de façon répétitive, sur une cible à l'aide d'un canon. Le principe du jeu est le suivant. La cible possède 250 points de forces. Chaque coup au but réduit les points de force de la cible. Lorsque la cible est détruite (lorsque ses points de force sont à 0), on détermine la capacité du joueur.

   Le déroulement du jeu est le suivant. Le joueur doit d'abord spécifier le type de canon qu'il désire utiliser. Cela consiste à spécifier la puissance des obus qu'il tire et la capacité du chargeur. Le joueur peut ensuite jouer plusieurs parties. Pour chaque partie, il doit spécifier la distance entre lui et la cible et le niveau de difficulté de la partie. Si la distance est négative, le jeu se termine. Le joueur peut ensuite effectuer une série de tirs qui se termine lorsque la cible est détruite ou lorsque le joueur abandonne. Pour effectuer ses tirs, il spécifie l'angle (en degrés) et la vélocité à laquelle le projectile est lancé. Si le projectile tombe à l'intérieur de la distance requise, c'est un coup au but. Si la cible est détruite, on affiche le résultat du joueur et on recommence.

   *Pour spécifier les types de canon, on doit spécifier la puissance des obus qu'il tire (de 15 à 75) ainsi que la capacité du chargeur (de 1 à 10 obus). Lors d'un coup au but, on se sert de la puissance d'un obus pour réduire les points de force de la cible (*points de force - puissance de l'obus*). La capacité du chargeur permet de tirer plusieurs obus en un seul essai. Entre chaque obus, le canon augmente de 1,5 degrés l'alignement du canon afin de couvrir une plus large surface.*

   Le jeu offre quatre niveaux de difficulté :

   (a) le niveau facile qui requiert que le projectile tombe à une distance de la cible qui est inférieure à 5 % de la distance du joueur de la cible ;

(b) le niveau moyen qui requiert que le projectile tombe à une distance de la cible qui est inférieure à 2 % de la distance du joueur de la cible ;

(c) le niveau difficile qui requiert que le projectile tombe à une distance de la cible qui est inférieure à 1 % de la distance du joueur de la cible ;

(d) le niveau expert qui requiert que le projectile tombe à une distance de la cible qui est inférieure à 0,5 % de la distance du joueur de la cible.

À la fin d'une partie, on calcule le pointage du joueur. Ce pointage est proportionnel au nombre d'essais requis pour détruire la cible (nombre de tirs) et la puissance de tir utilisé. Si on utilise une puissance totale d'au plus 250 à chaque essai, il n'y a aucune pénalité. Si on utilise une puissance entre 251 et 500, une pénalité de un essai est comptabilisée. Si on utilise une puissance supérieure à 500, une pénalité de deux essais est comptabilisée. À la fin, la qualité du joueur dépend du nombre d'essais normalisé qu'il a fait. Si le nombre d'essais normalisé est :

- *entre 1 et 8, la cote du joueur est « excellent » ;*
- *entre 9 et 15, le pointage est « très bon » ;*
- *entre 16 et 25, le pointage est « bon » ;*
- *entre 26 et 35, le pointage est « médiocre » ;*
- *entre 36 et 48, le pointage est « mauvais » ;*
- *plus de 49, la cote est « très mauvais ».*

La distance que parcourt un projectile, lancé avec un angle A (en radians) et une vélocité V (en pieds par seconde), avant de tomber au sol est donnée par la formule suivante :

$$distance = \frac{V^2 \times \sin(2 \times A)}{32.2}$$

Pour simplifier l'entrée des données, l'usager donne l'angle de tir en degrés. La formule pour convertir les degrés en radians est :

$$radian = \frac{degré \times PI}{180}$$

2. (35 points) **Sujet : L'album photos**

On vous demande de développer un logiciel permettant de gérer un album de photos. Le contenu de l'album est divisé en sections. On doit donc pouvoir ajouter et retirer des sections dans l'album.

Chacune des sections porte un nom unique et contient les photos. Le logiciel doit permettre d'ajouter et de retirer des photos dans les différentes sections. Les photos sont identifiées par un nom, une date, une catégorie (famille, visite, fête, tourisme, sport, ...) et le nom du fichier contenant la photo. On peut aussi associer un commentaire à chaque photo.

En plus des fonctions de base (ajouter/retirer), plusieurs autres actions sont possibles sur notre album. On peut vouloir consulter la liste des photos. Cette fonction peut s'appliquer sur l'album en entier ou bien une section particulière. Elle affiche tous les attributs associés aux photos sauf la photo elle-même. De plus, lors de l'affichage, on peut classer les photos par date ou catégorie.

On peut aussi vouloir visionner une photo. Ce visionnement se fait en sélectionnant le numéro associé à la photo suite à une consultation de la liste des photos.

Finalement, il doit être possible de visionner les photos dans un diaporama. Il permet de voir les photos dans un ordre spécifié par l'utilisateur (date ou catégorie) et à la vitesse de trois secondes par photo. Le diaporama peut concerner l'album en entier ou une section de l'album.

Faites l'analyse/conception orienté objet de ce problème. Donnez le diagramme de cas d'utilisation et les diagrammes de classes. Faites aussi le diagramme structurel et la conception de modules.

3. (15 points) **Sujet : La récursivité**

On vous demande d'écrire une fonction récursive permettant d'évaluer un nombre romain. Votre fonction reçoit en paramètres la chaîne de caractères représentant le nombre romain et sa longueur. La chaîne est représentée dans un tableau de caractères.

Pour représenter un nombre romain, les symboles utilisés ainsi que la valeur qui leur est associée sont :

M = 1000	D = 500	C = 100	L = 50	X = 10	V = 5	I = 1

Un nombre romain est donc une suite de chiffres romain. L'algorithme pour évaluer un nombre romain est assez simple. Soit un nombre romain représenté par une suite de chiffres romains (exemple DCLIV). Soit deux chiffres romains consécutifs dans cette suite. Si le premier chiffre a une valeur inférieure au suivant dans la liste, alors on le soustrait à la valeur totale du nombre, sinon on l'additionne à la valeur totale du nombre.

Voici quelques exemples de nombres romains :

- 2 s'écrit II ($2 = 1 + 1$)
- 3 s'écrit III ($3 = 1 + 1 + 1$)
- 4 s'écrit IV ($4 = -1 + 5$)
- 6 s'écrit VI ($6 = 5 + 1$)
- 7 s'écrit VII ($7 = 5 + 1 + 1$)
- 8 s'écrit VIII ($8 = 5 + 1 + 1$)
- 9 s'écrit IX ($9 = -1 + 10$)
- 15 s'écrit XV ($15 = 10 + 5$)
- 47 s'écrit XLVII ($47 = (-10 + 50) + 5 + 1 + 1$).
- 97 s'écrit XCVII ($97 = (-10 + 100) + 5 + 1 + 1$).
- 149 s'écrit CXLIX ($149 = 100 + (-10 + 50) + (-1 + 10)$)
- 1490 s'écrit MCDXC ($1490 = 1000 + (-100 + 500) + (-10 + 100)$)

Voici quelques remarques supplémentaires sur les nombres romains. Tous les nombres finissant par 1, 2 ou 3 se terminent dans l'écriture romaine par I, II ou III. La même remarque s'applique aux nombres terminant par 6, 7 ou 8. Ceci est valable aussi bien pour les unités, comme on vient de le voir, que pour les dizaines et les centaines ; ainsi 30 s'écrit XXX, 300 s'écrit CCC.

Tous les nombres finissant par 4, se terminent dans l'écriture romaine par IV. Tous les nombres finissant par 9, se terminent dans l'écriture romaine par IX. Ainsi 49 se termine par IX, et il s'écrit XLIX. Identiquement, les nombres finissant par 90 se terminent dans l'écriture romaine par XC.

4. (8 points) **Sujet : La complexité**

Combien de fois l'instruction `x++` est-elle exécutée dans le programme suivant ?

```
using namespace std;

int main()
{
   int i,k,x;

   cout << "Entrez la valeur de k : ";
   cin >> k;
   x=0;

   for (int j=0; j < k; j++)
   {
       i = 0;
       x++;
       do
       {    x++;
            i++;
       } while (i <= j);
   }
   cout << " x = " << x << endl;
}
```

Afin d'obtenir le maximum de points, vous devez détailler votre raisonnement et vos calculs.

5. (30 points) **Sujet : La page WEB du Père Noël**

Le Père Noël désire lancer un nouveau concept de choix cadeaux. Il veut que les personnes puissent choisir par Internet leur cadeau de Noël. Dans un premier temps, il désire offrir des jeux de constructions.

Le principe est que l'usager puisse choisir le nombre de pièces qu'il veut inclure dans sa commande, le type de chacune des pièces, leur taille ainsi que leur couleur.

Le jeu de construction est composé de morceaux de différentes formes et de différents matériaux. La taille des morceaux et leur couleur sont déterminées par le client. Les formes disponibles sont la sphère, le cylindre plein et le bloc rectangulaire (parallélépipède rectangle ou parrec). Les matériaux disponibles sont le bois, le plastique et le métal.

Dans un effort de rationalisation, le Père Noël désire transférer le coût du transport aux parents. Les parents devraient connaître ce coût de transport au moment de passer leur commande. Le coût de transport est déterminé selon le poids de chaque bloc. Ainsi, le bois et le plastique utilisés ont comme densité 0,5 kg au litre et le métal possède une densité de 1 kg au litre. Le coût du transport est de 3 $ du kilogramme (kg).

Finalement, les parents désirent aussi connaître le volume du paquet qui leur sera livré afin d'acheter suffisamment de papier d'emballage. Le volume du paquet est de 20 % supérieur au volume total des morceaux commandés.

Un usager, lorsqu'il place sa commande peut donc :

- ajouter un morceau de la forme, de la couleur et de la taille désirées ;
- lister les morceaux qui sont dans la commande ;
- établir le coût de transport ;
- établir le volume du paquet.

On vous demande donc d'écrire un programme implantant ce service d'achat en ligne. Vouc devez coder seulement les modules **Interface** et **ajouter morceaux** ainsi que les objets **Commande** et **sphere**.

Analyse globale du problème

Types utiles :

- Commande (tableau de morceaux de chaque type et nombre de morceaux de chaque type)
- Sphère (rayon)
- Parrec (trois côtés)
- Cylindre (rayon et hauteur)

Entrées :

- Suite d'opérations (caractères)
- Suite de morceaux (sphère, parrec, cylindre)
- Suite de couleurs (entier)
- Suite de matériel (caractères)

Sortie :

- Liste de morceaux
- Volume total
- Prix transport

Constantes :

- Commandes = a (ajouter) , l (lister), v (volume) et t (transport)
- Matériel = b (bois), p (plastique), m (métal)
- Densité matériaux : 0,5 kg/l et 1 kg/l
- Coût transport : 3 $ du kg
- PI = 3.14159
- Pondération du volume = 20 %

Formules :

- Volume = 4/3×PI×rayon^3 ou PI × rayon^2×hauteur ou cote1 × cote2 × cote3
- Transport = poids × coût transport
- Volume pondéré = volume × pondération du volume

Diagramme de cas d'utilisation

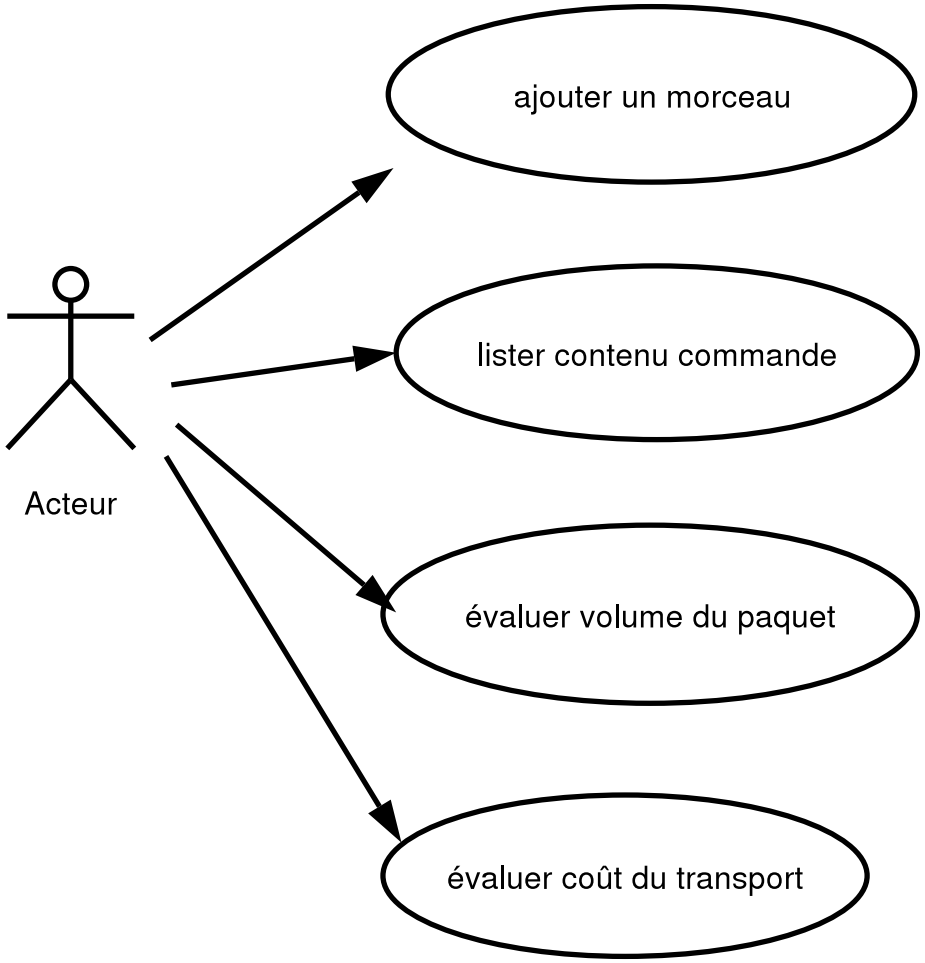
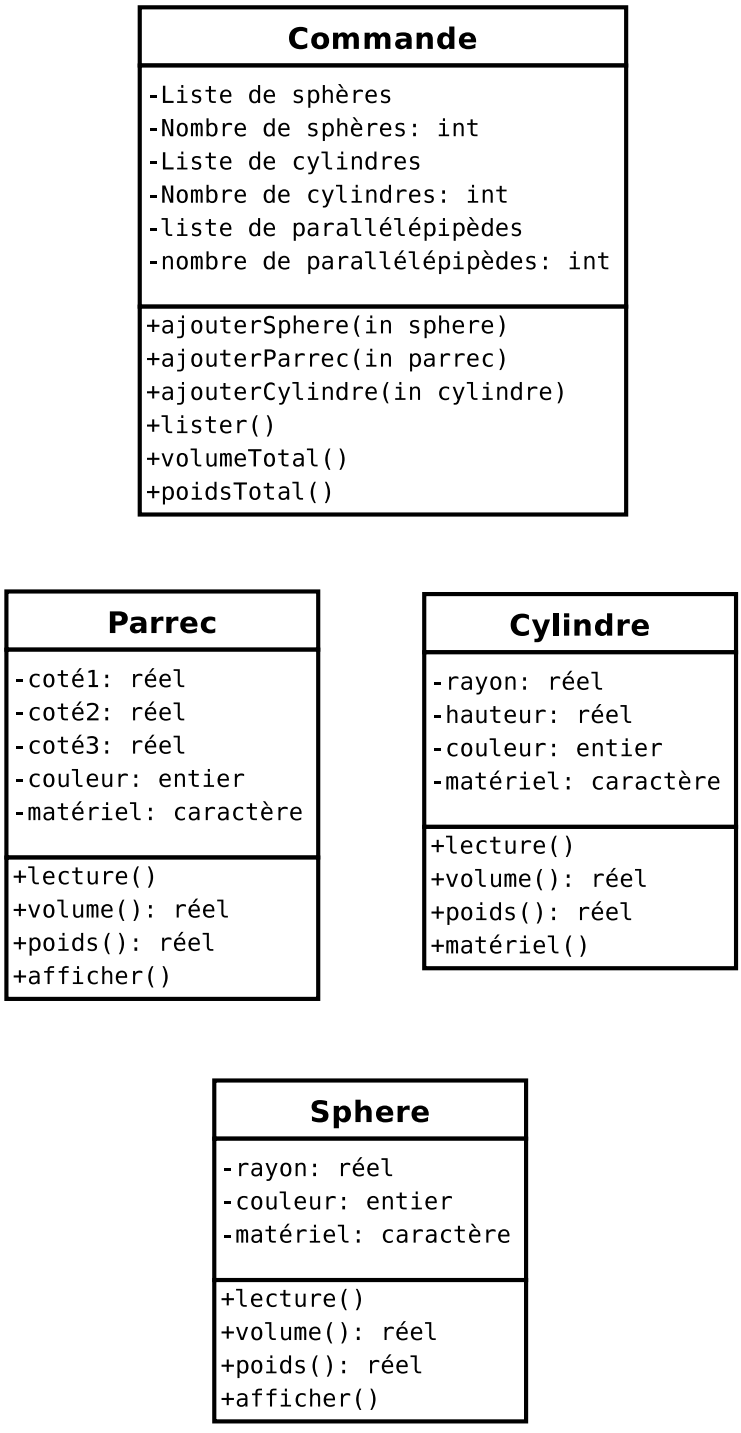


Diagramme de classes



MODULE *Interface (main)*

Entrée : (clavier) Opération (caractères)
Sorties : (écran) volume (réel)
(écran) coût du transport (réel)
Constantes : pondération du volume = 20 %
opération = a, l, v, t, f
coût transport : 3 $ du kg
Formules : volume pondéré = volume + (volume × pondération du volume)
transport = poids × coût transport

Conception

(a) Pour chaque opération (arrête sur f)

i. Lire opération

ii. Selon l'opération (a, l, v, t)

A. si ajouter — ajouter un morceau à la commande

B. si lister la commande — lister la commande

C. si coût transport
- calculer poids total de la commande
- calculer coût transport (formule)
- afficher coût du transport

D. si volume du paquet
- calculer volume total de la commande
- calculer volume pondéré (formule)
- afficher volume

MODULE *ajouter morceau*

Entrées : (paramètre) commande (commande)
(clavier) type de morceau (chaîne de caractères)
(clavier) le morceau (sphère, cylindre ou parrec)
Sortie : (paramètre) commande (commande)
Constantes : sphère, parrec, cylindre,
type de matériel : b, p, m

Conception

(a) Lire le type de morceau (sphère, parrec, cylindre)
(b) Selon le type de morceau
  i. si cylindre
    A. lecture cylindre
    B. ajouter le cylindre à la commande
  ii. si sphère
    A. lecture sphère
    B. ajouter la sphère à la commande
  iii. si parrec
    A. lecture parrec
    B. ajouter le parrec à la commande

Diagramme de séquence : Ajouter

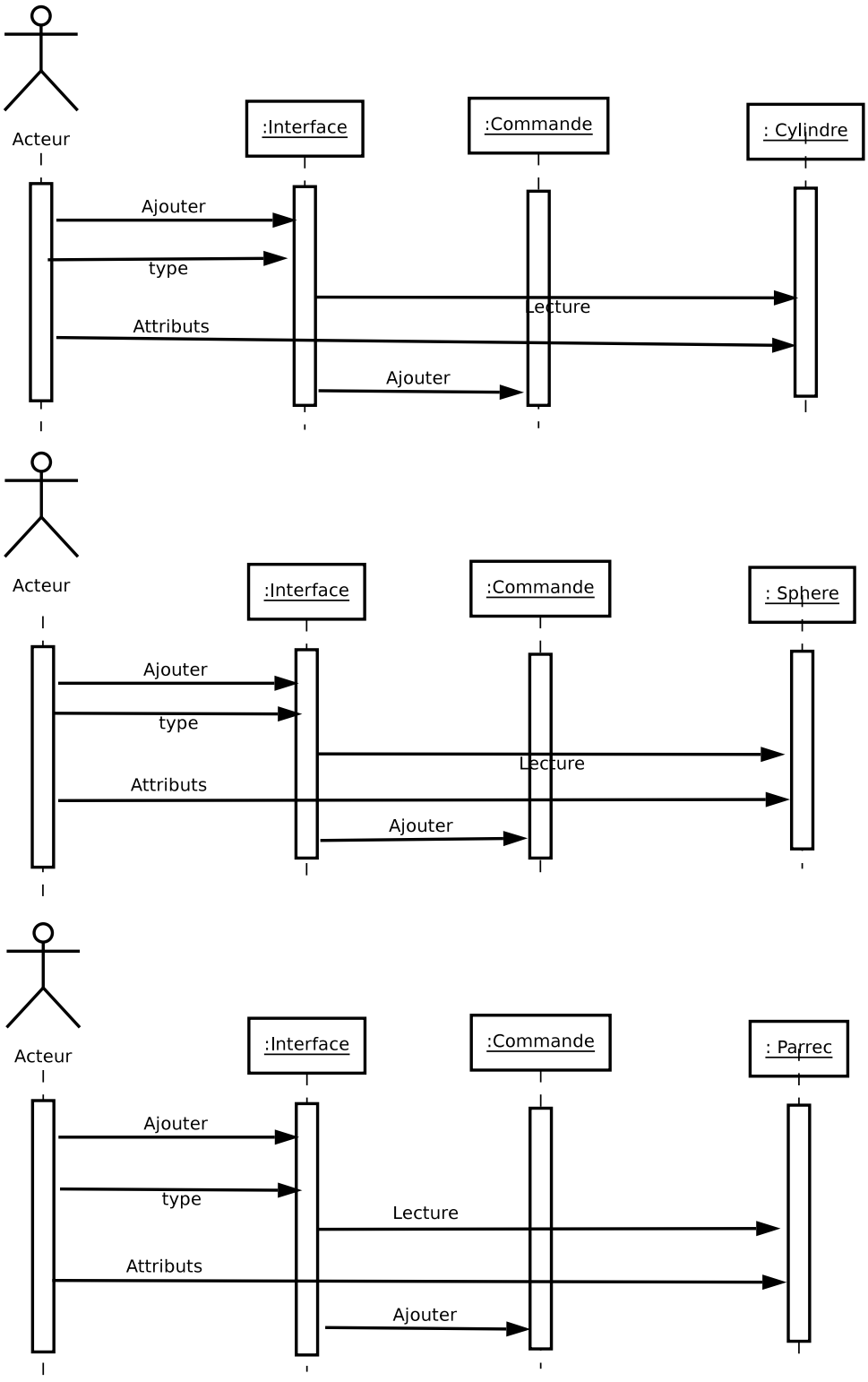
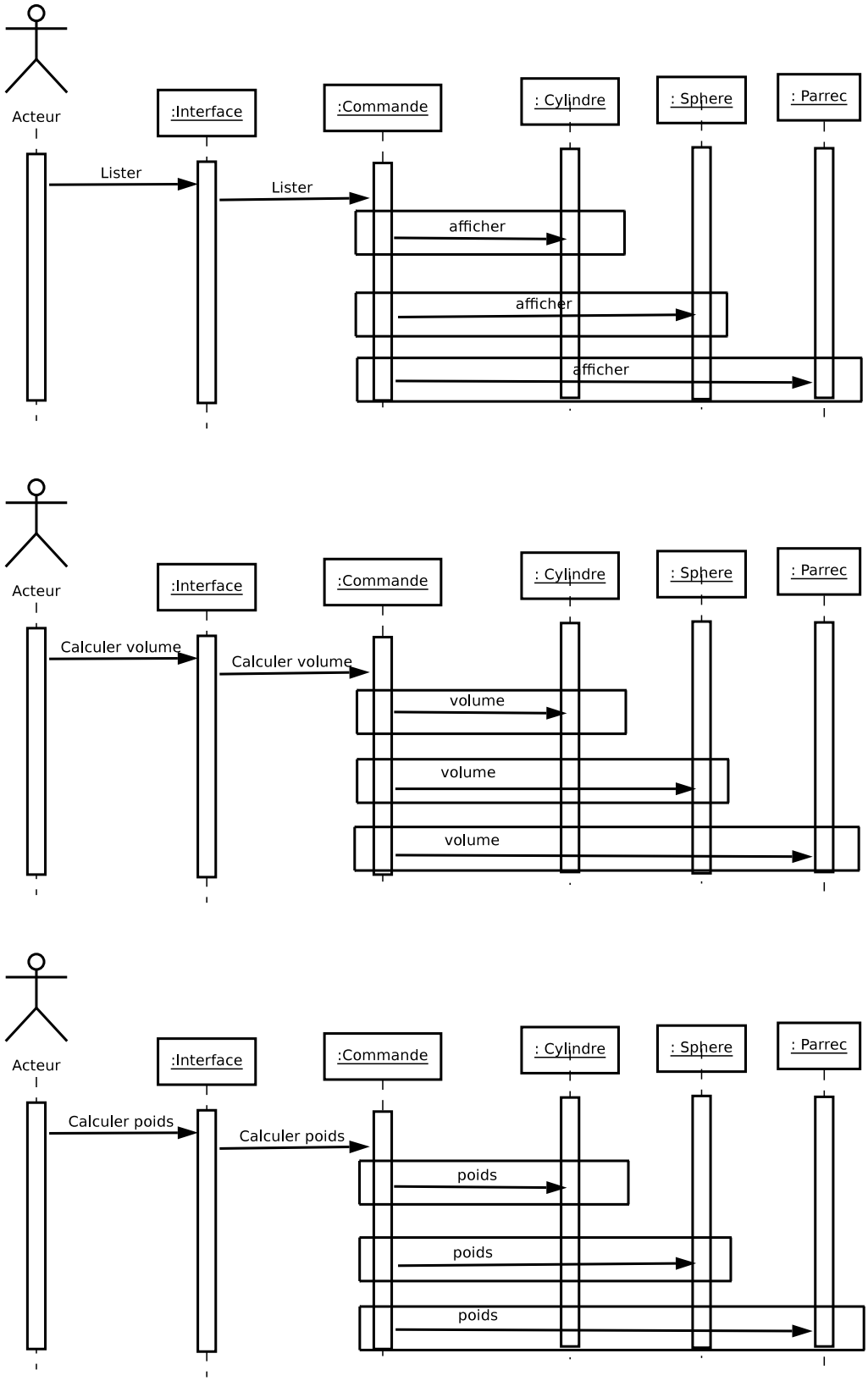


Diagramme de séquence : Lister, volume et poids



Type *Commande*

Méthode ajouterSphere

Entrées : (paramètre) une sphère (sphère)
(paramètre) commande (commande)

Sortie : (paramètre) commande (commande)

Conception (algorithme)

(a) Ajouter la sphère à la liste de sphère

(b) Incrémenter le nombre de sphères

Méthode ajouterParrec

Entrées : (paramètre) un parallélépipède (parrec)
(paramètre) commande (commande)

Sortie : (paramètre) commande (commande)

Conception (algorithme)

(a) Ajouter le parrec à la liste de parallélépipèdes

(b) Incrémenter le nombre de parrec

Méthode ajouterCylindre

Entrées : (paramètre) un cylindre (cylindre)
(paramètre) commande (commande)

Sortie : (paramètre) commande (commande)

Conception (algorithme)

(a) Ajouter le cylindre à la liste de cylindres

(b) Incrémenter le nombre de cylindres

Méthode lister

Entrée : (paramètre) commande (commande)

Sortie : (affichage) liste de morceaux (sphères, cylindres et parrec)

Conception (algorithme)

(a) Pour chaque liste (sphères, cylindres et parrec)

   i. pour chaque morceau

      A. afficher le morceau (sphères, cylindres et parrec)

Méthode calculer volume

Entrée : (paramètre) commande (commande)

Sortie : (paramètre) volume total (réel)

Conception (algorithme)

(a) Pour chaque liste (sphères, cylindres et parrec)

   i. pour chaque morceau

      A. calculer le volume du morceau (sphères, cylindres et parrec)

      B. cumuler les volumes

Méthode calculer poids

Entrée : (paramètre) commande (commande)

Sortie : (paramètre) poids total (réel)

Conception (algorithme)

(a) Pour chaque liste (sphères, cylindres et parrec)

   i. pour chaque morceau

      A. calculer le poids du morceau (sphères, cylindres et parrec)

      B. cumuler les poids

Type *Sphère*

Méthode lecture

Entrées : (clavier) rayon (réel)
(clavier) couleur (entier)
(clavier) matériel (caractère)
(clavier) sphère (sphère)

Sortie : (paramètre) sphère (sphère)

Conception (algorithme)

(a) Lecture du rayon, de la couleur et du matériel

Méthode volume

Entrée : (paramètre) sphère (sphère)

Sortie : (retour) volume (réel)

Formule : volume = 4/3 × PI × rayon^3

Constante : PI = 3.14159

Conception (algorithme)

(a) Calculer le volume (formule)

Méthode poids

Entrée :(paramètre) sphère (sphère)

Sortie : (paramètre) poids (réel)

Formule : poids = volumes × densité du matériel

Constantes : densité = 0,5 kg/litre ou 1 kg/litre
matériel = b (bois), p (plastique), m (métal)

Conception (algorithme)

(a) Calculer le volume (méthode volume)

(b) Selon le matériel

i. calculer le poids (formule)

Méthode afficher

Entrée :(paramètre) sphère (sphère)

Sortie : (affichage) sphère (sphère)

Conception (algorithme)

(a) Afficher tous les attributs

Type *Cylindre*

Méthode lecture

Entrées : (clavier) rayon (réel)
(clavier) hauteur (réel)
(clavier) couleur (entier)
(clavier) matériel (caractère)
(clavier) cylindre (cylindre)
Sortie : (paramètre) cylindre (cylindre)

Conception (algorithme)

(a) Lecture du rayon, de la hauteur, de la couleur et du matériel

Méthode volume

Entrée : (paramètre) cylindre(sphère)
Sortie : (retour) volume (réel)
Formule : volume = PI $\times$ rayon^2 $\times$ hauteur
Constante : PI = 3.14159

Conception (algorithme)

(a) Calculer le volume (formule)

Méthode poids

Entrée :(paramètre) cylindre (cylindre)
Sortie : (paramètre) poids (réel)
Formule : poids = volumes $\times$ densité du matériel
Constantes : densité = 0,5 kg/litre ou 1 kg/litre
matériel = b (bois), p (plastique), m (métal)

Conception (algorithme)

(a) Calculer le volume (méthode volume)
(b) Selon le matériel
    i. calculer le poids (formule)

Méthode afficher

Entrée : (paramètre) cylindre (cylindre)
Sortie : (affichage) cylindre (cylindre)

Conception (algorithme)

(a) Afficher tous les attributs

Type *Parrec*

Méthode lecture

Entrées : (clavier) trois côtés (réels)
(clavier) couleur (entier)
(clavier) matériel (caractère)
(clavier) parrec (parrec)
Sortie : (paramètre) parrec (parrec)

Conception (algorithme)

(a) Lecture des trois côtés, du rayon, de la couleur et du matériel

Méthode volume

Entrée : (paramètre) parrec (parrec)
Sortie : (retour) volume (réel)
Formule : volume = côté 1 × côté 2 × côté 3

Conception (algorithme)

(a) Calculer le volume (formule)

Méthode poids

Entrée :(paramètre) parrec (parrec)
Sortie : (paramètre) poids (réel)
Formule : poids = volumes × densité du matériel
Constantes : densité = 0,5 kg/litre ou 1 kg/litre
matériel = b (bois), p (plastique), m (métal)

Conception (algorithme)

(a) Calculer le volume (méthode volume)
(b) Selon le matériel
    i. calculer le poids (formule)

Méthode afficher

Entrée : (paramètre) parrec (parrec)
Sortie : (affichage) parrec (parrec)

Conception (algorithme)

(a) Afficher tous les attributs