

## TP GESTION DES MÉMOIRES

On voudrait réaliser une application orientée objet qui permet de déterminer les caractéristiques des mémoires informatiques les plus sollicités actuellement dans le marché des matériels informatiques, à savoir les mémoires vives (RAM), les mémoires Flash des clés USB, les disques durs, et les disques optiques (spécialement les DVD inscriptibles une seule fois de type DVD-R).

On définit une interface appelée **IMesurerMémoire** qui spécifie les méthodes suivantes :

- Capacité\_en\_Octets(char multiplicateur) qui convertit la capacité d'une mémoire en octets, ko, Mo, Go ou To, respectivement si la valeur du multiplicateur en argument est o, k, M, G ou T;
- Capacité\_en\_bits() qui convertit la capacité d'une mémoire en bits;
- 1 méthode Débit(char multiplicateur) correspondant au débit théorique d'une mémoire en bits/s, kb/s, Mb/s, Gb/s ou Tb/s, respectivement si la valeur du multiplicateur en argument est b, k, M, G ou T ;
- 1 méthode Débit\_en\_Octets(char multiplicateur) correspondant au débit théorique d'une mémoire en octets/s, ko/s, Mo/s, Go/s ou To/s, respectivement si la valeur du multiplicateur en argument est o, k, M, G ou T;

Rappel : **1 To = 1024 Go = 1024<sup>2</sup> Mo = 1024<sup>3</sup> ko = 1024<sup>4</sup> octets.**

On définit une interface appelée **IViewerMémoire** qui spécifie les méthodes suivantes permettant la préparation des mémoires pour accueillir des données:

- Formater(string système\_Fichiers) qui permet de vider le contenu d'une mémoire et la formater à un système passé en argument de la méthode;
- isRAW() qui indique si une mémoire n'est pas formatée;
- Nettoyer() permettant de diminuer l'espace occupé par les données de 10% si l'espace occupé dépasse 95% de la capacité de la mémoire.

On définit une interface appelée **IContrôlerMémoire** qui spécifie les méthodes suivantes permettant la gestion de l'utilisation des espaces mémoires :

- EspaceLibre() qui calcule l'espace vide de données d'une mémoire (en bits);
- EspaceLibre\_en\_Octets() qui calcule l'espace vide de données d'une mémoire (en octets);

- PourcentageUtilisation() qui détermine le pourcentage de l'espace occupé d'une mémoire;

On définit une classe abstraite **Mémoire**, qui possède :

- 2 variables d'instance : référence (de type **string**), capacité (exprimée en Go);
- 1 constructeur d'initialisation ;
- 2 accesseurs : Référence { **set**; **get**; } qui permet de retourner ou modifier la référence d'une mémoire, et Capacité { **get**; } qui permet de retourner la capacité d'une mémoire (en Go);
- 1 méthode abstraite isVolatile() qui indique si une mémoire est volatile;
- 1 méthode ToString().

On définit une classe abstraite **MémoireMasse**, représentant les mémoires de masse, qui hérite de la classe **Mémoire** et qui possède :

- 2 variables d'instance : système\_Fichiers (de type **string**) avec valeur par défaut **"RAW"**, et espaceOccupé (exprimé en Go) avec valeur par défaut 0;
- 2 constructeurs d'initialisation ;
- 2 accesseurs : Système\_Fichiers { **get**; } qui permet de retourner le système de fichiers utilisé dans une mémoire de masse, et EspaceOccupé { **get**; **set**; } qui permet de retourner ou modifier la taille de l'espace utilisé dans une mémoire de masse);
- 1 méthode abstraite isRéinscriptible() qui indique si les données d'une mémoire peuvent être effacées à volonté;
- 1 méthode ToString().

On définit une classe **MémoireVive**, représentant les mémoires vives, qui hérite de la classe **Mémoire** et qui implémente l'interface **IMesurerMémoire**, et qui possède en plus :

- 2 variables d'instance : fréquence qui indique la fréquence de fonctionnement d'une mémoire (en Hz), et largeur\_Bus qui indique la largeur du bus mémoire (en bits) ;
- 1 constructeur d'initialisation ;
- 2 accesseurs pour fréquence et largeur\_Bus.
- 1 méthode ToString().

On définit une classe `Flash_USB`, représentant les mémoires Flash des clés USB, qui hérite de la classe `MémoireMasse` et qui implémente les interfaces `IMesurerMémoire`, `IViderMémoire` et `IContrôlerMémoire`, et qui possède en plus :

- 1 variable d'instance : version (de type double) qui est soit 1.0, 1.1, 2.0, ou 3.0 (avec un débit de transfert de données maximal respectivement de 1,5 Mb/s, 12 Mb/s, 480 Mb/s et 5 Gb/s),
- 2 constructeurs d'initialisation ;
- 1 accesseur ;
- 1 méthode `ToString()`.

On définit une classe `DisqueDur`, représentant les disques durs, qui hérite de la classe `MémoireMasse` et qui implémente l'interface `IMesurerMémoire`, `IViderMémoire` et `IContrôlerMémoire`, et qui possède en plus :

- 3 variables d'instance :
  - capacité\_constructeur (déterminée par le fabricant du disque dur en Go, avec 1 Go = 1,000,000,000 octets),
  - bus (de type `string`) qui est soit "IDE" ou "SATA" (avec un débit de transfert de données maximal de 133Mo/s pour IDE et 600 Mo/s pour SATA),
  - nombrePartitions avec 1 comme valeur par défaut, qui permet de déterminer le nombre de partitions dans un disque dur;
- 3 constructeurs d'initialisation (la capacité est calculé automatiquement par conversion de la capacité déterminée par le fabricant vers l'unité de mémoire normalisée en Go),
- 2 accesseurs pour capacité\_constructeur et bus, et 1 accesseur NombrePartitions { `get`; };
- 1 méthode Partitionner(`int` nombre) qui permet de partitionner un disque dur selon le nombre passé en argument, le nombre doit être > 1;
- 1 méthode SupprimerPartitions() qui supprime toutes les partitions d'un disque dur;
- 1 méthode `ToString()`.

Un DVD a une capacité d 4,7Go et son système de fichiers s'appelle UDF (Universal Disk Format). On définit une classe `DVD_R`, représentant les DVD-R, qui hérite de la classe `MémoireMasse` et qui implémente l'interface `IParamétrerMémoire` et `IGérerMémoire`, (l'interface `IViderMémoire` est totalement inutile pour la classe `DVD_R`, car on ne peut pas formater un DVD-R ni supprimer ses données), et qui possède en plus :

- 3 variables d'instance : Vitesse\_lecture, Vitesse\_gravure (de type `int`) et multisession (de type `bool`) qui indique si on peut graver le disque sur plusieurs sessions ou non. La vitesse d'un DVD notée 1x correspond à un débit de 1385 ko/s. Par exemple, si la vitesse de lecture d'un DVD est 8x, son débit sera  $8 * 1385$  ko/s.
- 3 constructeurs d'initialisation,
- 1 accesseur pour multisession, et 2 accesseurs en lecture seule pour Vitesse\_lecture et Vitesse\_gravure;
- 1 méthode `isVierge()` qui permet de déterminer si un DVD-R est vierge;
- 1 méthode `ToString()`.

**Travail à réaliser :**

- a. Écrire les différentes interfaces et classes.
- b. Créer un jeu de tests avec 2 instances pour chacune des classes.