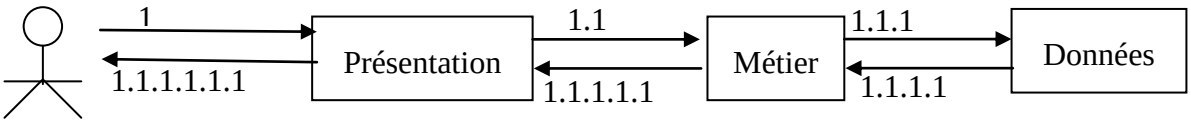
TD3 : Architecture 3-niveaux

Dans l'**architecture à trois niveaux** ou **architecture à trois couches** (3-tiers en anglais), la logique du système est divisée en trois niveaux ou couches.

Couche Présentation correspond à l'interface avec laquelle l'utilisateur interagit, nommé IHM (GUI en anglais). Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons…). Ces différents événements sont envoyés à la couche "métier". Elle n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par la couche "métier".

Couche Métier. Elle correspond à la partie fonctionnelle de l'application, elle implémente la « logique », et elle contient les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation. Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

Couche Données. Elle contient les données manipulées par l'application. Dans le cas typique d'une base de données, c'est le modèle qui la contient. Le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données.

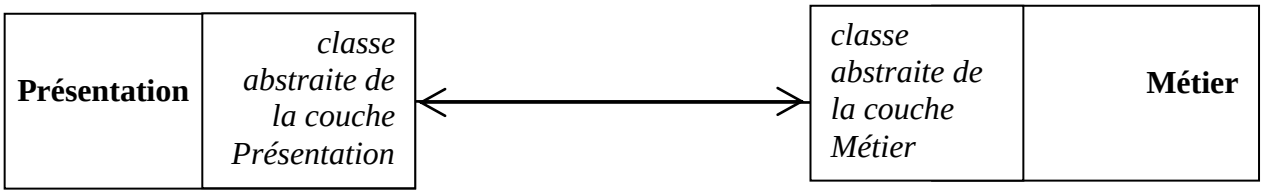


L'architecture 3-niveaux permet de créer des applications flexibles et réutilisables. Par la séparation des applications en niveaux, il est possible de modifier ou d'ajouter une couche spécifique, ou lieu de refaire l'application entière. Par exemple, modifier la couche présentation pour de nouvelle technologie telle qu'Android, IOS, Windows Phone, ...

Comparaison de l'architecture en couches avec les variantes de MVC

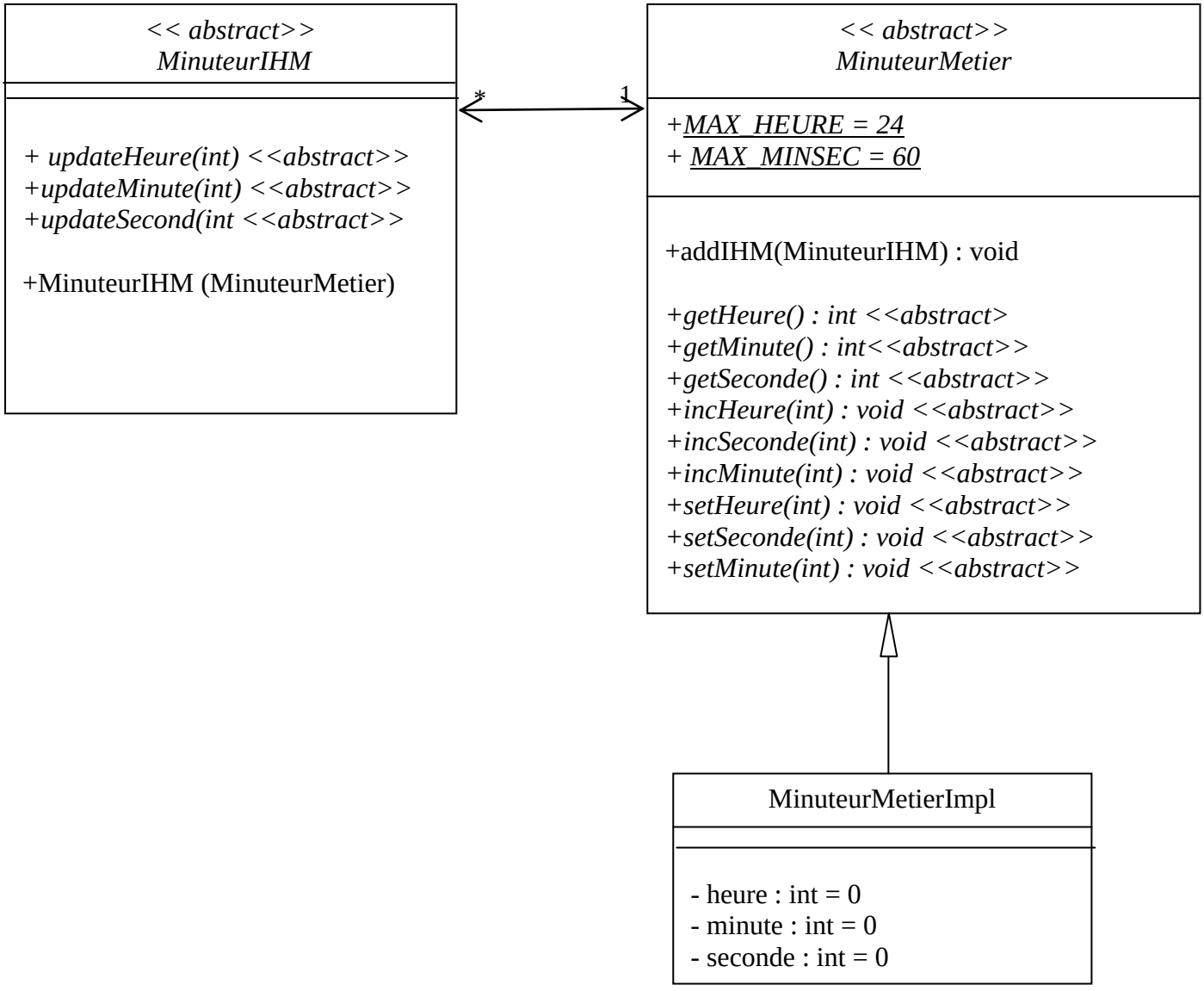
Dans l'architecture en couches, chaque couche communique seulement avec ses couches adjacentes (supérieures et inférieures) et le flux de contrôle traverse le système de haut en bas puis de bas en haut. Les couches supérieures contrôlent les couches inférieures, c'est-à-dire, que les couches supérieures sont toujours sources d'interaction (clients) alors que les couches inférieures ne font que répondre à des requêtes (serveurs). Dans le modèle MVC, il admit que la vue puisse consulter directement le modèle (en lecture) sans passer par le contrôleur.

L'objectif de ce TD est de concevoir pour une application spécifique les 2 classes abstraites Java définissant les interactions entre la couche Présentation et la couche Métier



Une fois, ces deux classes abstraites proprement établies la couche Présentation et la couche Métier sont découplées. Toutes les interactions entre la couche Métier avec la couche Présentation sont réalisées via la classe abstraite de la couche Présentation. Similairement, toutes les interactions entre la couche Présentation avec la couche Métier sont réalisées via la classe abstraite de la couche Métier.

Le contrôle d'un minuteur (heure, minute, seconde) via plusieurs interfaces réalisé par une application à 3 niveaux



Question : établissez une ébauche du diagramme de communication lors que l'utilisateur « demande incrémentation d'un seconde au moment où la valeur du Minuteur est 23h 59 min 59 secondes ».

Bataille navale

Comme son nom l'indique, la bataille navale est un jeu consistant à torpiller les navires de son adversaire. La flotte de chaque joueur est placée aléatoirement au début du jeu par l'application. Elle est composée :

* 1 porte-avion (5 cases consécutives, non diagonal)
* 2 croiseurs (4 cases consécutives, non diagonal)
* 2 sous-marins (3 cases consécutives)
* 2 torpilleurs (2 cases consécutives).

Ensuite, tour à tour, les joueurs lancent virtuellement des torpilles sur l'ennemi en indiquant les coordonnées d'un tir (par exemple B4). L'application indique si l'un de ses bateaux a été atteint ou pas. Le gagnant est celui qui parvient à torpiller complètement les navires de l'adversaire avant que tous les siens ne le soient.

La taille des grilles de jeu est toujours la même 10x10.

Chaque joueur voit une grille correspondant à la connaissance qu'il a acquise de la flotte de son adversaire : une case sans pion est une case non torpillée, un pion bleu indique une torpille qui n'a pas touché un navire ; un pion rouge indique une touche.

1. Listez les événements externes et les événements résultats.
2. Complétez la classe abstraite `BatailleNavaleMetier` et l'interface `BatailleNavalePresentation` : listez les méthodes nécessaires au bon fonctionnement du logiciel.
3. Donnez la signature des méthodes.
4. Etablir le diagramme de classes de la partie gestion du jeu.

Remarque : l'interface `BatailleNavaleMetier` est la façade de la partie « gestion du jeu ».
Information : sur la page Web de Colette Johnen (LaBRI) vous trouverez une implémentation en Java de la partie IHM.

Nom : **Prénom:** **Groupe:**