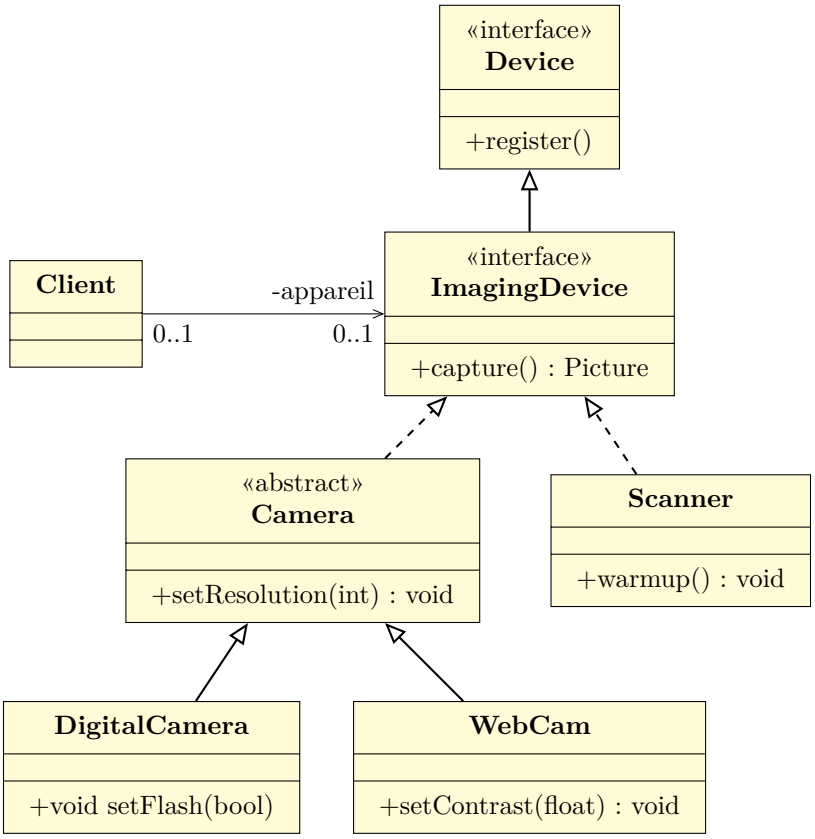
UML / Principes de POO

Exercice 1 – *Staticus, statica, staticum*

Soit le diagramme de classes suivant :



Question 1.1 : Listez toutes les entités de ce diagramme en donnant leur type.
Question 1.2 : Comment se traduit, en Java, l'association `appareil` (écrivez le code de la classe `Client`) ? Même question si cette association avait une cardinalité `0..*`.
Question 1.3 : Rappelez la distinction entre type statique et type dynamique et à quoi ils s'appliquent ?
Question 1.4 : Sur ce diagramme, où trouve-t-on le type statique de la variable `appareil`. Énumérez également les types dynamiques potentiels visibles sur ce diagramme.
Question 1.5 : Donnez tous les diagrammes d'objets correspondant aux associations possibles dénotées par ce diagramme de classes.
Question 1.6 : Dans chaque configuration d'objets, précisez la liste des méthodes que `Client` peut appeler à partir de `appareil`.

Exercice 2 – *Principes de base*

Question 2.1 : Récupérez le contenu du dossier `poly` de l'archive et importez-le dans un nouveau projet `Eclipse`.
Question 2.2 : Donnez une modélisation UML complète de ce programme. Indications : il devrait y avoir 3 paquetages, 3 entités et 2 relations.

Question 2.3 : Énumérez les principes vus en cours que ce programme et son architecture pourraient enfreindre. On souhaite ajouter une classe `Monome` dont l'intérêt est de permettre une représentation plus efficace puisqu'un monôme [1] n'a besoin de mémoriser que son degré et son unique coefficient là où `Polynome` doit stocker dans un tableau tous les coefficients (même nuls) jusqu'à son degré.

Pour cela, il faudra que la classe `Monome` puisse être traitée exactement de la même manière qu'un `Polynome`. C'est-à-dire que partout où un `Polynome` est attendu, un `Monome` pourra être donné à la place sans que le code qui l'utilise en ait connaissance.

Question 2.4 : Quelle relation doit exister entre les types `Polynome` et `Monome` ?

Question 2.5 : Comment peut-on établir cette relation sans modifier le code existant de la bibliothèque ? Est-ce satisfaisant ? Correct ?

Question 2.6 : Proposez une architecture qui corrige tous les problèmes rencontrés tout en conservant toutes les fonctionnalités de l'architecture précédente. Donnez un diagramme UML complet de cette architecture.

Question 2.7 : Une fois vérifiée, programmez la nouvelle architecture. Vous modifierez le programme de test pour que le polynôme `p2` soit généré à partir de monômes plutôt qu'à partir d'une liste de coefficients.

Question 2.8 : Bonus : si vous souhaitez étudier un cas où un sous-typage par héritage se justifie, on pourrait imaginer l'ajout d'une classe `PolynomeRange` qui reprend l'implémentation de `PolynomeImpl` en évitant de mémoriser les premiers coefficients dès lors qu'ils sont nuls. Pour cela, on pourra ajouter un attribut pour mémoriser le degré du premier coefficient non nul et s'appuyer sur la classe de base pour le reste.

Question 2.9 : Bonus : En vous inspirant d'un exemple vu en cours, quelle opération ne peut être ajoutée au type `Polynome` sans invalider la relation de sous-typage entre `Monome` et `Polynome` ?

1. Un monôme est un polynôme à un seul coefficient non-nul. Il est de la forme $a \cdot x^n$.