Génie Logiciel

Diagramme de séquence

TD/TP4

Séverine Affeldt, Stanislas Morbieu

MLDS - LIPADE
UFR Mathématiques et Informatique

Université Paris Descartes

Le diagramme de séquence

Le diagramme de séquence établit un **lien** entre le diagramme de **cas d'utilisation**, où les acteurs interagissent avec les grandes fonctionnalités du système, et le diagramme de **classes**, qui décrit le coeur du système.

Le diagramme de séquence apporte un aspect **dynamique** à la modélisation du système. Il se focalise sur l'**échange** d'information entre les éléments. Les participants à un échange sont appelés **lignes de vie**. Une ligne de vie représente un participant unique à un échange. Le terme "ligne de vie" est utilisé car au cours d'un échange, des objets peuvent être créés, utilisés et parfois détruits

Le diagramme de séquence montre des interactions sous un angle **temporel**. Plus précisemment, il donne le séquencement temporel des messages échangés entre des lignes de vie.

Le diagramme de séquence en détail

Différent types de messages:

- l'envoi d'un message
- l'invocation d'une opération
- la création ou la destruction d'une instance

L'**Envoi d'un messsage** déclenche une réaction chez le récepteur, de manière asynchrone et sans réponse. L'emetteur du message n'est pas bloqué pendant l'envoi et ne sait pas si/quand le message sera traité.

L'**Invocation d'une opération** correspond en POO à `c.op()`, où `c` est une instance de la classe `C` et `op()` est une méthode. La plupart des invocations sont synchrones (sinon on utilise un *thread*).

Avec UML, l'envoie de message et l'invocation se représentent de la même manière. On peut toutefois faire la différence entre un message synchrone et un message asynchrone.

Exercice 1

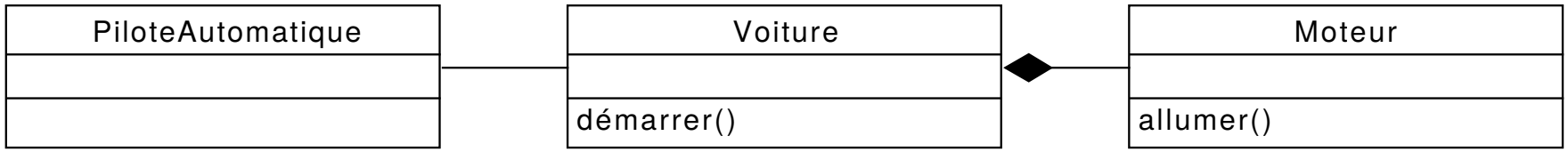
Donnez le diagramme de séquence correspondant à l'envoi d'un message `introduireCarte` à une classe `Distributeur` par une client. Représentez le début et la fin d'exécution par le `Distributeur` sur la ligne de vie.

Exercice 2

Donnez le diagramme de séquence correspondant à l'invocation d'une méthode `chercher(Livre)` pour une classe `Médiathèque`. Cette invocation implique le renvoi d'un nombre de 'Livre' non nul.

Exercice 3

Le diagramme de classes ci-dessous donne la structure interne d'un système de pilotage automatique composé de trois classes: `PiloteAutomatique`, `Voiture` et `Moteur`.



Proposez un diagramme de séquence modélisant les interactions entre ces classes.

Exercice 4

On souhaite modéliser la diagramme de séquence correspondant au retrait d'argent via un distributeur de billets. Les deux classes possibles sont `Distributeur` et `Réseau`.

Le distributeur permet au client d'introduire sa carte (`introduireCarte`) et de saisir son code (`saisieCode(code)`). Il permet également l'affichage de l'écran de saisie (`afficherEcranSaisieCode`).

Le réseau de la banque vérifie le code envoyé par le distributeur (`vérifierCode(code)`) et renvoie une information de vérification (`banqueOK=vérifierCode(code)`).

Exercice 5

On considère deux classes: `Pilote` et `Avion`. Une instance de la classe `Pilote` peut invoquer les méthodes `démarrerMoteur()` et `décoller()` d'une instance de la classe `Avion`. L'invocation de `démarrerMoteur()` nécessite la vérification par le classe `Avion` du niveau d'essence.

Modélisez ce fonctionnement par un diagramme de séquence.

Exercice 6

Lors d'un retrait de billets au distributeur (classe `:Distributeur`), le client (classe `:Client`) a le choix entre des euros ou des dollars. Le distributeur affiche alors ou bien un écran en francais, ou bien un écran en anglais.

Modélisez ce fonctionnement par un diagramme de séquence.

Exercice 7

Avant le démarrage d'un train (`:Train`), le conducteur (`:Conducteur`) demande la fermeture des portes (`fermerPortes()`). Les n portes (`:Porte`) sont alors fermées (`fermer()`).

Modélisez ce fonctionnement par un diagramme de séquence comportant une boucle.

Exercice 8

On considère le travail d'un contrôleur aérien.

Un pilote vérifie la check-list de son avion (le diagramme correspondant appelé `controlerCheckList` n'est pas décrit en détail mais simplement référencé). La check-list est transmise en argument car elle est utilisée par l'interaction `controlerCheckList`. Cette interaction retourne un booléen qui est affecté à l'attribut ok de l'avion.

Lorsque le contrôleur demande à contrôler l'avion, le booléen ok est testé. Si l'avion est prêt à partir, son plan de vol est validé; dans le cas contraire, le plan de vol est invalidé.

L'interaction `contrôleAvion` recoit en argument une instance de `CheckList`. Cet argument est en entrée/sortie (inout) car il est utilisé dans le diagramme et pourra servir encore dans un autre diagramme qui référencera `contrôleAvion`.

`contrôleAvion` retourne un résultat du type `PlanDeVol`. L'instance correspondante est la ligne de vie appelée planDeVol.

Modélisez ce fonctionnement par un diagramme de séquence.