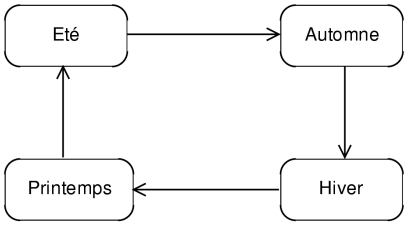
TD6 : Diagramme d'états/transitions
UML

Pierre Gérard
pierre.gerard@univ-paris13.fr

DUT Informatique S2D
Université de Paris 13

1 Transitions « change »et « after »

Le changement de saisons constitue une boucle continue. On considère un objet de la classe « Saison »de durée de vie infinie. En utilisant des événements de type « change »ou « after », donner le diagramme des états-transitions de la classe Saison correspondant aux états de l'année climatique de la France (printemps, été, automne, hiver).



Les événements sont when(date=21 mars), when(date=21 juin)... ou alors after(3 mois)

2 Etats associés à une classe

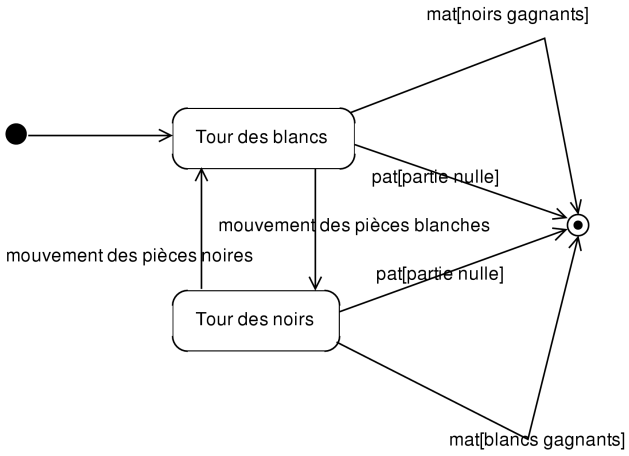
Considérons une classe Partie dont la responsabilité est de gérer le déroulement d'une partie de jeu d'échecs. Cette classe peut être dans deux états :

- le tour des blancs
- le tour des noirs.

Les événements à prendre en considération sont

- un déplacement de pièces de la part du joueur noir
- un déplacement de pièces de la part du joueur blanc
- la demande de prise en compte d'un échec et mat par un joueur. S'il est validé par la classe partie, un échec et mat assure la victoire du dernier joueur. Dans ce cas, une activité « noirsGagnants »ou « blancsGagnants »– selon le cas – est déclenchée (appel de méthode).
- la demande de prise en compte d'un pat qui mène aussi à une fin de partie, avec une égalité. Dans ce cas, une activité « égalité »est déclenchée.

Question : Donner le diagramme d'états/transitions associé à la classe Partie.



3 Diagrammes et sous-diagrammes

Une montre digitale propose une fonction horloge et une fonction chronomètre. Elle est munie de quatre boutons, l'appui sur chacun des boutons donnant lieu à un événement :

- Le bouton « light », quand il est pressé, allume une lumière. La lumière reste allumée pendant une durée de deux secondes, après quoi elle s'éteint à nouveau. Un appui sur ce bouton alors que la lumière réinitialise la durée l'allumage de la lumière.
- Le bouton « mode »active successivement les trois modes principaux de la montre : l'affichage de l'heure courante, le mode chronomètre et le mode réglage (pour mettre à jour l'heure courante).
- En mode chronomètre, le bouton « start/stop »lance et arrête le chronomètre. En mode réglage, il active successivement les fonctions de réglage de l'heure, des minutes ou des secondes.
- En mode réglage, le bouton « set »incrémente les heures, les minutes ou les secondes. En mode chronomètre, il remet le chronomètre à zéro.

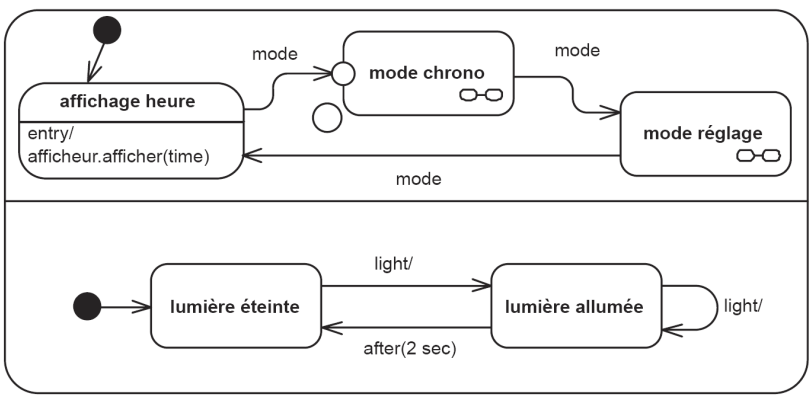
Nous allons décomposer l'étude du fonctionnement de cette montre en plusieurs étapes, en établissement des diagrammes d'états/transition concernant la classe montre.

3.1 Etats composites

L'affichage est géré par un objet ayant le rôle d'« afficheur »et disposant d'une opération afficher qui permet d'afficher la chaîne de caractères qu'on lui donne en paramètre. Le décompte du temps est géré par un objet ayant le rôle de « time »et disposant d'une opération toString retournant une chaîne de caractères correspondant à l'heure courante.

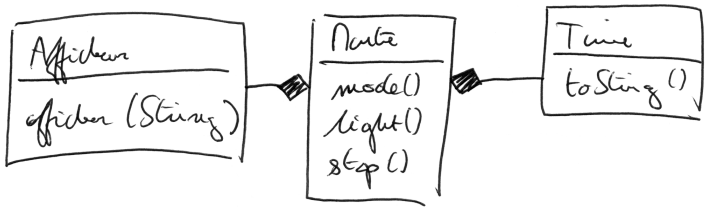
Question : Modélisez dans un premier temps le comportement lié à l'utilisation du bouton « mode »de la montre.

Question : Dans le même diagramme, ajoutez le comportement lié à l'utilisation du bouton « light ».



Dans affichage heure : do/afficheur.afficher(time.toString())

Question : Donnez aussi un diagramme de classes pour modéliser les relations et les opérations des objets de type « Afficheur », « Time »et « Montre ».



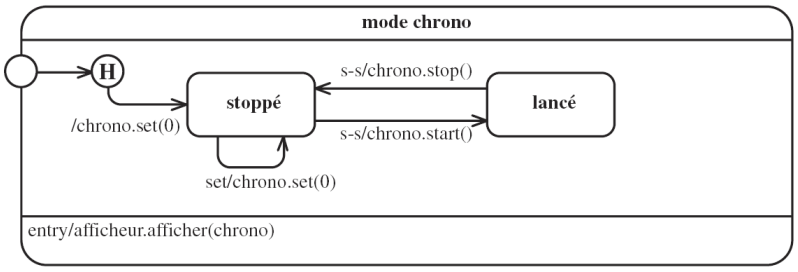
3.2 Etats composés

En mode chronomètre, le bouton « start-stop »lance ou arrête le chronomètre. Le bouton « set »remet le chronomètre à zéro s'il est arrêté.

Question : Modélisez ces comportements.

Il est en fait possible de lancer le chronomètre, puis de basculer en mode affichage de l'heure et de revenir au mode chronomètre. Dans ce cas, le chronomètre continue de tourner pendant cette opération.

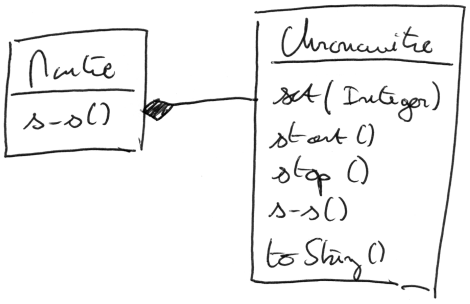
Question : Ajoutez cette possibilité de retrouver l'état précédent en changeant de mode. Vous pourrez utiliser un pseudo-état historique.



do/afficheur.afficher(chrono.toString())

Le décompte du temps chronométré est géré par un objet avec le rôle de « chrono »et qui dispose des opérations stop, start, init et enfin toString retournant une chaîne de caractères.

Question : Ajoutez une classe « Chronomètre »au diagramme de classes.

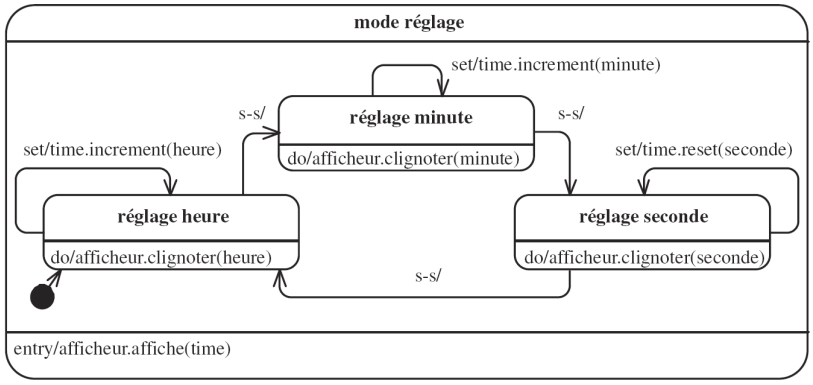


L'afficheur dispose de méthodes indicateurHeure, indicateurMinute et indicateurSeconde pour afficher l'indicateur adéquat. La classe « Time »dispose de méthodes incrementeHeure, incrementeMinute et initSeconde.

En mode réglage, le bouton start-stop active successivement les fonctions de réglage de l'heure, des minutes ou des secondes. Selon la fonction de réglage courante, l'afficheur affiche un indicateur sous les heures, les minutes ou les secondes. Le bouton set incrémente l'heure courante d'une heure en mode

réglage de l'heure, d'une minute en mode réglage des minutes, ou remet les secondes à zéro en mode réglage des secondes. L'heure affichée ne change plus dès qu'on entre dans le mode réglage.

Question : Modélisez ces comportements à l'aide d'un diagramme d'états-transitions.



Dans affichage heure : do/afficheur.afficher(time.toString())

Question : Enrichissez encore le diagramme de classes de manière à ce qu'il soit cohérent avec le diagramme d'états/transitions.