Analyse UML et Codage C++

Projet Horloge sous Linux

~ Séquence n°1 ~

Version 3.0 – janvier 2019

Objectif

- Analyse UML
  - diagramme de cas d'utilisation
  - diagramme de séquence
  - diagramme états-transitions
  - diagramme de classes
  - Notion d'acteur
  - Notion de classe
  - Relation entre classes

Conditions de réalisation

Ressources utilisées :

Matériel	**Logiciel**
Un PC sous linux	Modelio

1. PRÉSENTATION DU SYSTÈME

Ce projet permet, l'étude d'une horloge. Elle indique les heures et les minutes sous la forme **hh mm** sur un cadran. Un clavier avec trois touches <Mode>, <Plus> et <Moins> permet d'effectuer les différents réglages :

- Un premier appui sur la touche <Mode> permet le réglage des heures et produit l'affichage **HH hh**, les touches <Plus> et <Moins> font alors évoluer l'heure.
- Un second appui produit l'affichage **MM mm** et permet le réglage des minutes dans les mêmes conditions.

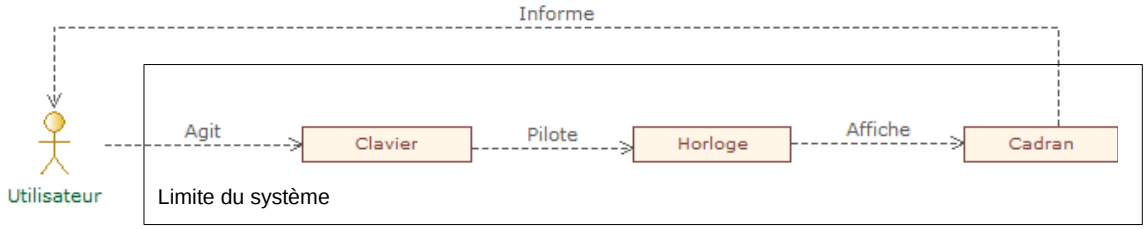
Les valeurs **hh** et **mm** représentent ici des valeurs numériques.

- Enfin un troisième appui sur la touche <Mode> revient au mode de fonctionnement classique de l'horloge.
- Pour le bon fonctionnement du système, on peut prévoir une touche <FIN> pour terminer le programme lors de la phase de mise au point.

L'horloge peut avoir une résolution sur 12 ou 24 heures, cette option est définie à la fabrication du logiciel (sans entraîner des modifications importantes dans le code).

2. ANALYSE

2.1. *Diagramme de contexte*



Le clavier utilise les touches [Espace] pour changer de mode, [+] et [-] pour le réglage des heures et des minutes. La touche [Entrée] peut être utilisée pour terminer le programme.

Le Cadran dispose d'une seule ligne d'affichage.

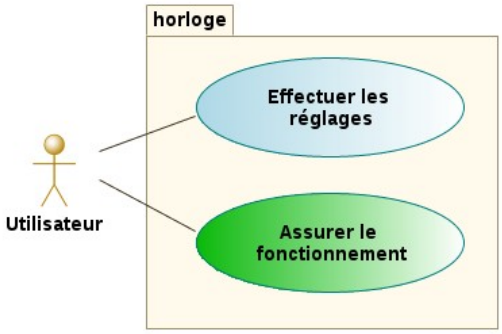
2.2. *Catalogue des acteurs*

NOM	DESCRIPTION	CAS D'UTILISATION
Utilisateur	*Dispose d'une horloge pour obtenir l'heure courante. Cette horloge peut être réglée à l'aide de trois touches.*	Effectuer les réglages Assurer le fonctionnement

2.3. Modèle des cas d'utilisation

Les cas d'utilisation pour ce projet sont au nombre de deux :

- Effectuer les réglages
- Assurer le fonctionnement



2.4. Détail des cas d'utilisation

Détail du cas d'utilisation : 'Assurer Fonctionnement'

Utilisé par :

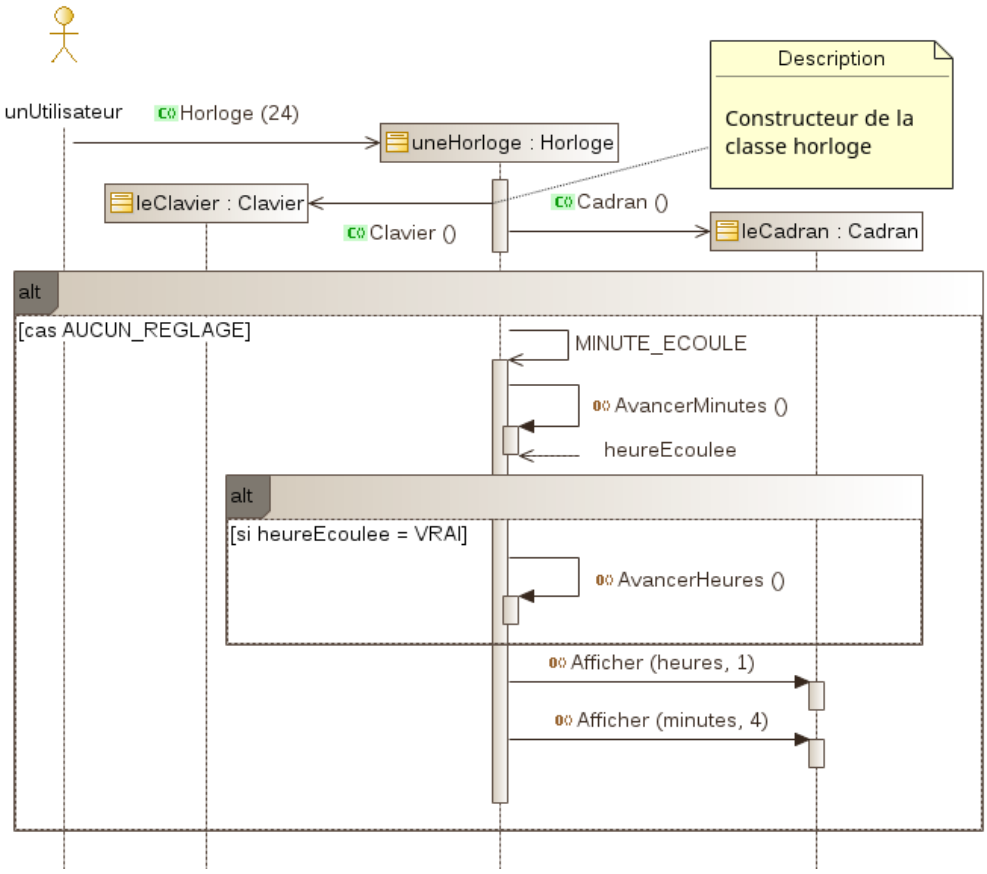
[Acteurs]

Utilisateur

Description :

Le lancement de l'horloge active le Clavier et l'écran. Chaque minute écoulée dans le mode aucun réglage, le calcul de l'heure est actualisé. La valeur des heures et des minutes est affichée sur le cadran.

Diagramme de Séquence :



Détail du cas d'utilisation : 'Effectuer les réglages'

Utilisé par :

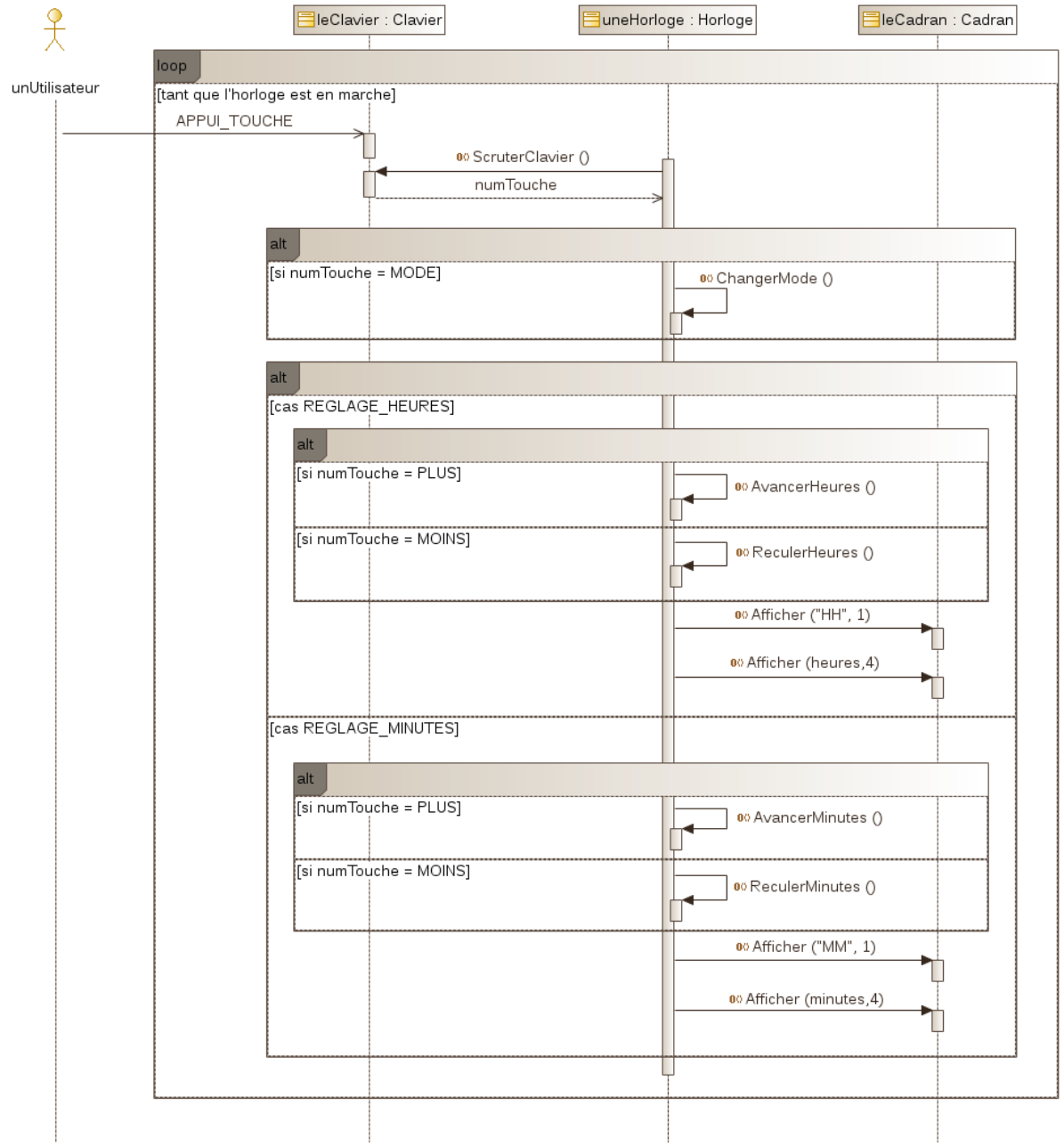
[Acteurs]

Utilisateur

Description :

Lorsque le clavier est sollicité par un appui sur la touche MODE, l'horloge passe alternativement en réglage des heures, réglage des minutes ou en mode aucun réglage. En fonction du mode dans lequel se trouve l'horloge, la prise en compte des touches PLUS et MOINS est réalisée pour effectuer le réglage des heures ou le réglage des minutes. Pour chaque mode, un affichage spécifique est proposé à l'utilisateur.

Diagramme de Séquence :



3. TRAVAUX DIRIGÉS

3.1. Complément du dossier d'analyse

1. À partir des diagrammes de séquence, complétez les tableaux suivants :

Nom des instances utilisées	Nom de Classes
UneHorloge	Horloge
LeClavier	Clavier
leCadran	Cadran

Messages synchrones	Émetteur	Récepteur
APPUI_TOUCHE	Utilisateur	Clavier
MINUTE_ECOULEE	Horloge	Horloge

Nom des méthodes	Nom de la classe à laquelle elles appartiennent	Paramètre de retour (si oui → Type)
ScruterClavier	Clavier	TOUCHES_CLAVIER
ChangerMode	Horloge	
AvancerHeures	Horloge	
ReculerHeures	Horloge	
Afficher	Cadran	
Afficher	Cadran	
AvancerMinutes	Horloge	Booléen (HeureEcoulee)
ReculerMinutes	Horloge	

2. Indiquer les attributs à ajouter dans la classe ***Horloge*** en précisant leur type de donnée respectif ?

Attribut	**Type**
heures	entier
minutes	entier
resolution	
mode	

3. Après avoir observé l'extrait de code de la classe ***Cadran*** et en utilisant les éléments étudiés sur la classe ***Horloge*** et la classe ***Clavier***, ébaucher le diagramme de classe de l'application. Quelles relations existe-t-il entre ces différentes classes ? Justifier votre réponse en vous aidant des diagrammes présentés dans ce document.

   Composition entre Horloge et Cadran

   Composition entre Horloge et Clavier
   La classe Horloge crée le Clavier et le Cadran et appelle les méthodes de chacune des classes

4. En étudiant l'extrait de code de la classe Cadran, identifier le constructeur, le destructeur (indiquer le rôle de chacun d'eux d'une manière générale) À quel instant sont-ils respectivement appelés ?

   Le constructeur initialise les attributs et sauvegarde le contexte de l'application (par exemple)

   le destructeur restitue la contexte.

   Le constructeur est appelé lorsque la classe est in

5. Qu'apportent les mot-clé ***private*** et ***public*** dans le fichier ***cadran.h*** ?

6. Le constructeur possède des valeurs par défaut pour ces différents paramètres. Quelle est l'utilité de ces valeurs par défaut ?

   Permet d'initialiser les paramètres sans fixer explicitement une valeur.
   Quelles sont les valeurs par défaut pour un cadran ?

   _posX= 1 ; _posY =1 _hauteur=1 ; _largeur= 7

7. Que signifie le mot clé ***const*** dans le passage de paramètre des fonctions membres ?

   Pour indiquer que la paramètre est constant, il ne peut pas être modifié dans la fonction (notion paramètre d'entrée)

8. Comment peut-on justifier le fait qu'il existe dans la classe ***Cadran*** deux méthodes ***Afficher( )*** ?

   Les paramètres sont différents
   Comment se nomme cette particularité du C++ ?

   La surcharge d'opération

9. En utilisant l'atelier de génie logiciel Modelio, réalisez le diagramme de cas d'utilisation, le diagramme de classes et le diagramme états-transitions. Enfin, réalisez le premier diagramme de séquence « Assurer le fonctionnement ».

```
/**
 * @file   Cadran.cpp
 * @author  Philippe CRUCHET
 * @date 31 décembre 2018, 16:09
 */

#include "Cadran.h"
/**
 * Rôle :   Constructeur de la classe, Initialise le cadran
 *          et trace le contour.
 * @param _posX     position en X coin haut-gauche du cadran
 * @param _posY     position en Y coin haut-gauche du cadran
 * @param _hauteur  nombre de lignes d'affichage
 * @param _largeur  nombre de caractères par ligne
 */
Cadran::Cadran(const int _posX, const int _posY, const int _hauteur, const int _largeur) :
posX(_posX),
posY(_posY),
hauteur(_hauteur),
largeur(_largeur)
{
   initscr();
   keypad(stdscr, TRUE);
   noecho();
   mvaddch(posY, posX, '+');
   mvaddch(posY, posX + largeur + 1, '+');
   mvaddch(posY + hauteur + 1, posX, '+');
   mvaddch(posY + hauteur + 1, posX + largeur + 1, '+');
   mvhline(posY, posX + 1, '-', largeur);
   mvhline(posY + hauteur + 1, posX + 1, '-', largeur);
   mvvline(posY + 1, posX, '|', hauteur);
   mvvline(posY + 1, posX + largeur + 1, '|', hauteur);
   curs_set(0);
   refresh();
}

/**
 * Rôle : Destructeur de la classe
 */
Cadran::~Cadran()
{
   endwin();
}

/**
 * Rôle : Affiche un texte à la position donnée
 * @param texte     Chaine de caractères à afficher
 * @param position  position du 1er caractère, 0 = 1re position
 */
void Cadran::Afficher(const char *texte, const int position)
{
   mvprintw(posY + 1, posX + position + 1, texte);
   refresh();
}
```

```
/**
 * Rôle : Affiche un entier sur 2 chiffres à la position donnés
 * @param valeur    entier à afficher
 * @param position  position de la dizaine, 0 = 1re position
 */
void Cadran::Afficher(const int valeur, const int position)
{
   char *texte = new char[largeur + 1];
   if (texte != NULL)
   {
      // à compléter
      Afficher(texte, position);
      delete texte;
   }
}

/**
 * Rôle : Efface la zone d'affichage de l'écran
 */
void Cadran::Effacer()
{
   mvhline(posY + 1, posX + 1, ' ', largeur);
   refresh();
}
```

```
/**
 * @file:   Cadran.h
 * @author: Philippe
 *
 * @date:   31 décembre 2018, 16:09
 */
#ifndef CADRAN_H
#define CADRAN_H

#include <ncurses.h>

class Cadran {
public:
   Cadran(const int _posX=1,const int _posY=1,const int _hauteur=1,const int _largeur=7);
   virtual ~Cadran();
   void Afficher(const char *texte, const int position=0);
   void Afficher(const int valeur,const int position=0);
   void Effacer();

private:
   int posX;
   int posY;
   int hauteur;
   int largeur;

};

#endif /* CADRAN_H */
```