

Analyse UML et Codage C++

Projet Horloge sous Linux

~ Séquence n°3 ~

Version 3.0 – janvier 2019

Objectif

- Codage C++
 - Constructeur
 - Atributs
 - Méthodes
 - Constantes énumérées
 - Codage diagramme états-transitions
- UML
 - Relations entre classes : composition

Conditions de réalisation

Ressources utilisées :

Matériel

Un PC sous linux

Logiciel

NetBeans
Modelio

1. SUITE DE L'EXTRAIT DU DOSSIER D'ANALYSEDescription des classes**Classe : 'Horloge'****Description :**

Cet objet gère l'heure courante et dispose d'opérations pour la mise à l'heure.

Attributs : [Privé]

heures short	Heure courante, valeur comprise entre 00 et 11 ou entre 00 et 23 suivant la résolution.
minutes short	Minute courante, valeur comprise entre 00 et 59
resolution short (constant)	Résolution de l'horloge sur 12 ou 24 heures (par défaut 24)
mode TMODE	Définit le mode de fonctionnement de l'horloge à savoir : AUCUN_REGLAGE = 0, REGLAGE_HEURES, REGLAGE_MINUTE.
nbModes short (constant)	Nombre de modes possible par défaut 3

Opérations : [Public]

Horloge (in short <i>_nbMode</i> , in short <i>_resolution</i>) par défaut <i>_resolution</i> = 24	
	Constructeur de la classe, initialise les attributs heures et minutes à 0, nbMode et resolution en fonction des paramètres reçus et mode à la valeur AUCUN_REGLAGE. Il crée également l'instance du Clavier et du Cadran.
~Horloge ()	
	Destructeur de la classe, libère la mémoire des instances du Clavier et du Cadran.
boolean AvancerHeures ()	
	Passe à l'heure suivante en tenant compte de la résolution . Retourne vrai si passage à la demi-journée suivante ou au jour suivant en fonction de la résolution.
boolean AvancerMinutes ()	
	Passe à la minute suivante. Retourne vrai si passage à l'heure suivante.
void ReculerHeures ()	
	Revient à l'heure précédente.
void ReculerMinutes ()	
	Revient à la minute précédente.
TOUCHE_CLAVIER Controler (in TOUCHES_CLAVIER <i>_numTouche</i>) par défaut <i>_numTouche</i> = 0	
	Assure le fonctionnement de l'horloge pour le mode dans lequel elle se trouve. Elle reçoit le numéro de la touche enfoncée, agit en accord avec le diagramme états-transitions de la classe Horloge. Assure les affichages correspondant au mode et retourne numéro de la touche enfoncée après avoir scruter le clavier à la fin de la méthode.
void ChangerMode ()	
	Passe au mode suivant en tenant compte du nombre de nbModes

Classe : 'Clavier'

Description :

Cet objet gère les actions de l'utilisateur sur le clavier

Attributs : [Privé]

etatInitial struct termios	Sauvegarde de l'état antérieur du clavier.
-----------------------------------	--

Opérations : [Public]

Clavier ()	
	Constructeur de la classe, sauvegarde la configuration antérieure du clavier et le paramètre en mode lecture non bloquante
~Clavier ()	
	Destructeur de la classe, restitue l'état initial du clavier
TOUCHES_CLAVIER ScruterClavier ()	
	Lit le clavier du système, si aucune touche n'est enfoncée retourne AUCUNE (la valeur 0), MODE pour la touche [ESPACE], PLUS pour la touche [+], MOINS pour la touche [-] et FIN pour la touche [Entrée]. Pour toutes les autres touches du clavier la fonction retourne AUCUNE.

2. CODAGE DU CLAVIER EN C++

1. Réalisez un nouveau projet nommé TestClavier de type application C++ sous NetBeans. Ajoutez les fichiers **clavier.h** et **clavier.cpp** à ce projet.
2. Dans le fichier **clavier.h**, complétez la déclaration de la classe Clavier par la description de la constante énumérée **TOUCHES_CLAVIERS** comme le montre la figure suivante :

```
enum TOUCHES_CLAVIER
{
    AUCUNE,
    FIN,
    MODE,
    PLUS,
    MOINS
};
```

3. Dans le fichier **clavier.cpp**, complétez la méthode ScruterClavier pour qu'elle retourne le numéro de la touche enfoncée en respectant les indications données dans la description de la classe.
4. Complétez le programme principal pour qu'il affiche le numéro de la touche enfoncée si elle est différente de la valeur AUCUNE. Le programme s'arrête lorsque l'utilisateur appui sur la touche correspondant à FIN. Donnez la valeur de chaque constante.

3. CODAGE DE HORLOGE EN C++

5. Réalisez un nouveau projet nommé **Horloge** de type application C++ sous NetBeans. Ajoutez les fichiers **clavier.h**, **clavier.cpp**, **cadran.h** et **cadran.cpp** à ce projet. Ce projet doit pouvoir utiliser la librairie ncurses puisqu'il utilise le Cadran étudié lors de la séquence n°2.
6. Créez une nouvelle classe nommée Horloge, ajoutez à la déclaration une constante énumérée contenant les 3 modes de l'horloge.

7. Poursuivez la déclaration de la classe par la déclaration des opérations dans la section publique de la classe et des attributs dans la section privée.

Ajoutez un attribut de type **time_t** de la librairie **time.h** nommé **valAvant**, il sera utilisé pour mémoriser l'heure précédente pour savoir si une minute est écoulée.

Prévoyez également les deux pointeurs sur le Clavier et le Cadran pour coder la relation entre les classes.

8. Codez le constructeur et le destructeur de la classe en respectant la description proposée précédemment. l'opérateur **new** réalise l'allocation dynamique de mémoire et **delete** assure la destruction de l'instance.

Complément pour le codage :

<code>leCadran = new Cadran(5,5);</code>	Initialisation de l'instance du Cadran à la position 5,5. Le nombre de lignes est fixé avec la valeur par défaut, de même pour la largeur du cadran.
<code>delete leCadran;</code>	Détruit l'instance du Cadran, et libère la mémoire allouée dynamiquement.
<code>valAvant = time(NULL);</code>	Initialisation de l'attribut avec la valeur courante de l'heure.

Vous utiliserez la liste d'initialisation pour les autres attributs de la classe.

Complétez par la définition des autres méthodes sans pour autant les coder.

9. Dans le programme principal, instanciez la classe Horloge de manière automatique comme le montre la figure suivante :

```
int main(int argc, char** argv) {
    Horloge uneHorloge ;
```

Le cadran doit apparaître dans votre console au lancement du programme

10. Procédez au codage des méthodes **AvancerHeures**, **AvancerMinutes**, **ReculerHeures**, **ReculerMinutes** en respectant les descriptions fournis.
11. La méthode **ActualiserHeure** est donnée ici. Elle permet de n'avancer les minutes que si 60 secondes sont écoulées.

```
void Horloge::ActualiserHeure()
{
    time_t valCourante = time(NULL);
    double seconde = difftime(valCourante, valAvant);

    if (seconde > 60) // à modifier pour aller plus vite pendant le test
    {
        valAvant = valCourante;
        if (AvancerMinutes())
            AvancerHeures();
    }
}
```

12. Codez la méthode **ChangerMode** en utilisant l'opérateur modulo '%' en langage C.
13. En respectant le diagramme états-transitions et en tenant compte des deux diagrammes de séquence, réalisez le codage de la méthode **Controler**. Elle assure le cœur du fonctionnement de l'horloge.
14. Pour tester votre horloge, vous pourrez utiliser le programme principal suivant :

```
int main(int argc, char** argv) {
    Horloge uneHorloge ;

    TOUCHES_CLAVIER laTouche = AUCUNE;

    do
    {
        laTouche = uneHorloge.Controler(laTouche);
    } while (laTouche != FIN);

    return 0;
}
```