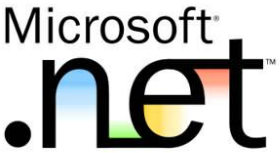
## Enoncé du laboratoire

# Application programmation orientée objet

Bacheliers en Informatique et systèmes
Bacheliers en Informatique de gestion
2ème année




Cours de théorie donné par **Alfonso Romio**

Séances de laboratoire dispensées par :

**François Caprasse**
**Cécile Moitroux**
**Alfonso Romio**
**Patrick Quettier**

**Année académique 2013-2014**

# Table des matières

1. Introduction ..... 3
2. Fonctionnalités ..... 4
Phase 1 : Evaluation la semaine du 17/02/2014 ..... 4
Phase 2 : Evaluation la semaine du 17/03/2014 ..... 7
3. Phase 3 : Application finale ..... 9
Mise en situation et objectifs ..... 9
Architecture ..... 9
Les composants de l'application ..... 10
Fonctionnalités ..... 13
A définir avant de commencer à développer ..... 15
4. Planning ..... 16
5. Contenu du dossier final ..... 17
6. Règles d'évaluation ..... 18

# 1. Introduction

Ce document présente l'ensemble des technologies à utiliser dans le cadre du laboratoire de "Application programmation orientée objet".

De façon générale, il s'agira de créer une application permettant la simulation de la consommation d'énergie dans un bâtiment.

Pour préparer cette étape, les développements seront divisés en 3 parties :

1) Développement de classes et outils de base : Dossier et Evaluation : Semaine du 17 février 2014
2) Application "Dessin d'objets graphiques" : Dossier et Evaluation : Semaine du 17 mars 2014
3) Application "Optimize Energy consumption" : Dossier et Evaluation : Session d'examen de Juin 2014

**Les fonctionnalités à développer et l'architecture à mettre en place sont précisées de façon très claire et sont à respecter.** En cas de retard, des sanctions seront prises.

L'énoncé est à réaliser individuellement. A chaque évaluation, les étudiants délivreront, un fichier archive contenant:

- Les solutions (sans les fichiers exécutables)
- Un rapport détaillé (description disponible sous peu sur l'école virtuelle)

Le nom du fichier et l'objet du mail seront obligatoirement constitués de la manière suivante :

*APOO Groupe Nom.rar*

Celui-ci sera envoyé par mail à l'adresse du titulaire de laboratoire :
francois.caprasse@hepl.be ou
cecile.moitroux@hepl.be ou
alfonso.romio@hepl.be ou
patrick.quettier@hepl.be

au plus tard le matin de l'évaluation.

Les outils à utiliser pour les développements sont :

- Visual Studio 2010 Professional ou Visual Studio 2013 Professional (anglais)
- Framework .NET 3.5 au minimum

L'ensemble des outils seront installés et configurés **avant** la première séance de laboratoire.

# 2. Fonctionnalités

### *Phase 1 : Evaluation la semaine du 17/02/2014*

| | |
|---|---|
| **Classe Appareil** | Il s'agit de créer une classe **Appareil** permettant de réaliser l'ensemble des fonctionnalités proposées dans les différents modules de tests qui sont fournis en complément à cet énoncé. Ces tests ont été élaborés pour qu'in-fine votre classe **Appareil** comporte :<br>- Des variables membres<br>- Une variable de classe (statique)<br>- Des propriétés associées<br>- Une série de constructeurs<br>- Des méthodes classiques<br>- Une ou plusieurs méthode(s) statique(s)<br>- Des surcharges d'opérateurs<br>- Une ou plusieurs méthode(s) surchargée(s)<br>- Des méthodes nécessaires à l'implémentation de certaines interfaces clés<br>- Un indexeur |
| **Application console** | Il s'agit donc de créer un projet en mode console.<br>Dans celui-ci vous créerez la classe **Appareil** dont il est question ci-dessus. Il s'agira de remplacer le fichier Program.cs par celui qui est fourni. Ensuite chaque test décrit ci-dessous doit être validé l'un après l'autre. Pour cela il suffit de « dé-commenter » le ***define*** correspondant au numéro de test qui vous intéresse. |

# Description des Tests

**TEST 1 : Test des constructeurs et des propriétés de la classe**

Créer une classe ***Appareil*** possédant au minimum 3 constructeurs permettant :

- d'initialiser l'objet par défaut
- d'initialiser l'objet avec une marque et un modèle
- d'initialiser l'objet avec une marque, un modèle, un numéro de série, une consommation moyenne et un type permettant de catégoriser les appareils (Electroménager, Média, Chauffage, Eclairage, Autre) [utiliser une énumération]

Le numéro de série d'un appareil est généré automatiquement à partir d'une chaîne constante de base et d'un numéro incrémenté automatiquement lors de la création de chaque appareil.
Surcharger la méthode ToString ne prenant aucun paramètre.

Implémenter l'interface IFormattable afin d'avoir un affichage détaillé ou sommaire.

**TEST 2 : Test de l'égalité d'appareils à l'aide de la méthode « Equals »**

Il s'agit de créer une méthode permettant à un objet ***Appareil*** de se comparer avec un autre objet ***Appareil***. L'égalité porte sur l'égalité du numéro de série.

**TEST 3 : Test de l'égalité d'appareils à l'aide de l'opérateur ==**

Il s'agit de surcharger l'opérateur == de la classe ***Appareil***. Cela permettra de comparer deux objets ***Appareil*** en utilisant l'opérateur == classique. L'égalité porte sur le numéro de série.

**TEST 4 : Test de la présence d'un appareil dans une liste**

Les classes permettant de gérer des collections de données présentes au sein du framework .NET se basent sur les interfaces implémentées par les données qu'elles contiennent pour déterminer les fonctionnalités qu'elles peuvent réaliser. La comparaison de deux objets entre eux fait partie de ses comportements. Autrement dit, il est nécessaire d'implémenter l'interface adéquate au sein de la classe ***Appareil*** pour que la classe ***List<Appareil>*** soit capable de détecter la présence ou non d'un appareil « identique » avant d'en effectuer son insertion dans la liste.

**TEST 5 : Comparaison d'appareil via le tri dans une liste**

La liste template ***List<Appareil>*** peut être triée à l'aide de la méthode Sort. Cette méthode comporte plusieurs versions. La première, qui nous intéresse ici, permet de trier les objets ***Appareil*** à l'aide de la méthode ad hoc nécessairement présente au sein de la classe ***Appareil*** lorsqu'elle implémente l'interface dédicacée à la comparaison. On fera en sorte que cette méthode permette de trier les appareils de manière croissante à partir d'un tri effectué sur le numéro de série.

La dernière partie du test permet d'effectuer un tri en utilisant la deuxième version de la méthode ***Sort*** et un délégué. Elle nécessite l'utilisation d'un objet ***Comparison<Appareil>***

dont la raison d'être est de fournir la méthode de comparaison à utiliser. En l'occurrence, une méthode statique présente au sein de la classe ***Appareil*** dans le cas qui nous intéresse. Celle-ci trie les appareils sur base de leur date de mise en service.

**TEST 6 : Une classe dédicacée à la comparaison**

La méthode ***Sort*** de la classe ***List<Appareil>*** comporte une troisième version permettant de trier des joueurs à l'aide d'un objet définissant la manière dont la comparaison s'effectue. Il s'agit donc de créer une classe ***MyAppareilComparer*** implémentant l'interface adéquate pour effectuer la tâche demandée.

**TEST 7 : Test des indexeurs**

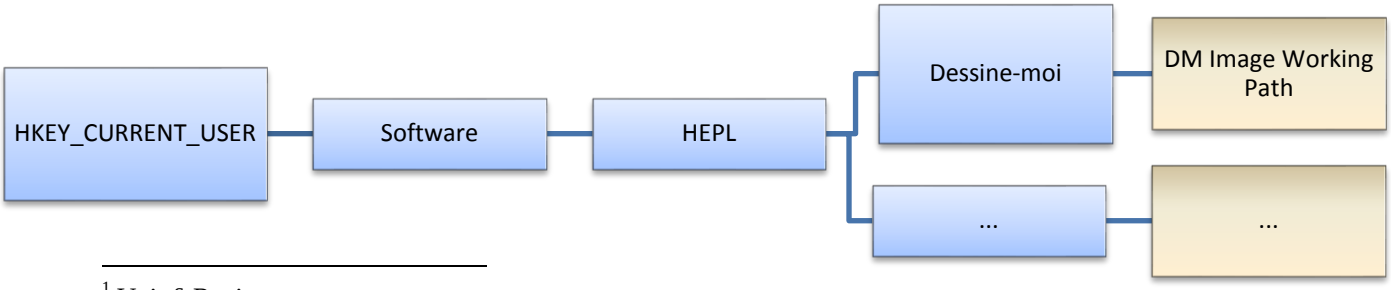
La classe ***Autorisation*** permet de gérer les plages horaires pendant lesquelles un appareil est en fonctionnement. Cette classe est membre privée de la classe ***Appareil*** par une relation d'agrégation par valeur. Les plages horaires sont arrondies à la demi-heure.
L'indexeur associé à la classe ***Autorisation*** permet de rapidement atteindre l'élément ad hoc pour l'initialiser ou le consulter. Key est une propriété qui retourne une chaîne de caractère composée de la marque, du modèle et du numéro de série de l'appareil. De par son contenu, elle est unique pour chaque appareil.

## *Phase 2 : Evaluation la semaine du 17/03/2014*

Il s'agit de développer une application permettant de dessiner dans une fenêtre graphique.

| | |
|---|---|
| Hiérarchie de classes | Les éléments "dessinables" sont des formes géométriques - simples ou complexes – et des images.<br><br>Les formes simples sont des cercles, des carrés (classe ***Square***), des rectangles (classe ***Rect***). Les formes complexes sont composées d'une liste de formes simples ou d'images<br><br>Suivant les cas, les éléments "dessinables" sont caractérisés par<br>• les dimensions des formes (mesure du côté du carré, rayon du cercle, ...),<br>• son point d'ancrage dans la fenêtre (Structure ***Point***),<br>• la couleur (classe ***Color***) utilisée pour le dessiner,<br>• le statut "rempli" (Filled) ou non,<br>• le nom du fichier image,<br>• un pourcentage représentant la mise à l'échelle de l'image,<br>• ...<br><br>Il s'agit donc de créer une hiérarchie de classes implémentant (certaines dans une hiérarchie contenant éventuellement une classe abstraite …) l'interface ***IDrawable***.<br><br>L'interface ***IDrawable*** contient plusieurs méthodes Draw(...) qui sont implémentées dans les classes implémentant cette interface. Elle offre également la propriété ***TopLeft*** qui permettra d'accéder au point d'ancrage de la forme ou de l'image. |
| Application Console | Créer une application console permettant de tester la hiérarchie de classes précédemment créée.<br><br>Dans le jeu de test proposé, insérer les différents objets dans une List<IDrawable> et appeler la méthode Draw() pour chacun des objets de la liste.<br><br>La méthode *void Draw()* de l'interface ***IDrawable*** affiche sous la forme de chaîne de caractères, les caractéristiques des formes créées. |

| | |
|---|---|
| Application Winform | Créer une application Winform "Dessine-moi" contenant une zone dans laquelle les éléments seront dessinés (objet ***PictureBox***).<br>Ajouter le code permettant d'ajouter des formes de caractéristiques différentes. Plusieurs méthodes sont possibles.<br>1. La forme est créée à partir de données encodées dans une ***PropertyGrid***.<br>2. Les formes sont créées à l'aide de clicks dans la ***PictureBox*** et de la commande Drag&Drop. Le click down désigne un des sommets de la forme. Le Click up désigne le second sommet. Les valeurs des coordonnées relevées permettent de déterminer les valeurs du TopLeft de la forme ainsi que ses dimensions.<br><br>Ajouter le code nécessaire à la création d'une forme complexe. L'édition de celle-ci se fera dans un formulaire prévu à cet effet. La fermeture du dit formulaire permettra de valider ou d'invalider la forme. En cas de validation, la forme (complexe) est alors ajoutée à la liste des éléments « dessinables » de la fenêtre principale et apparaitra dans la zone d'affichage prévue à cet effet.<br><br>Ajouter le code nécessaire à l'insertion d'une image choisie dans un dossier spécifique[1]. |
| **Registry** | Certains éléments des applications qui seront développées seront mémorisés par l'intermédiaire de la Registy. Celle-ci est disponible à partir du menu "démarrer" dans lequel on peut encoder "regedit" pour pouvoir exécuter l'éditeur qui permet de modifier ses données.<br><br>Dans votre application, la classe "Registry" définie dans le Framework .NET offre un ensemble de propriétés et méthodes qui permettent des opérations sur celle-ci.<br><br>Les éléments qui vont être mémorisés dans la registry sont organisés pour respecter la structure décrite ci-dessous.<br><br>Les dossiers et sous-dossiers (représentés en bleu) sont à créer au démarrage de l'application s'ils n'existent pas encore. Les éléments marqués en beige sont les noms des éléments auxquels sont associées des valeurs sous la forme de chaînes de caractères.<br><br>DM Image Working Path = répertoire contenant les images de l'application "Dessine-moi".<br><br>D'autres données seront ajoutées plus tard. |




[1] Voir § Registry

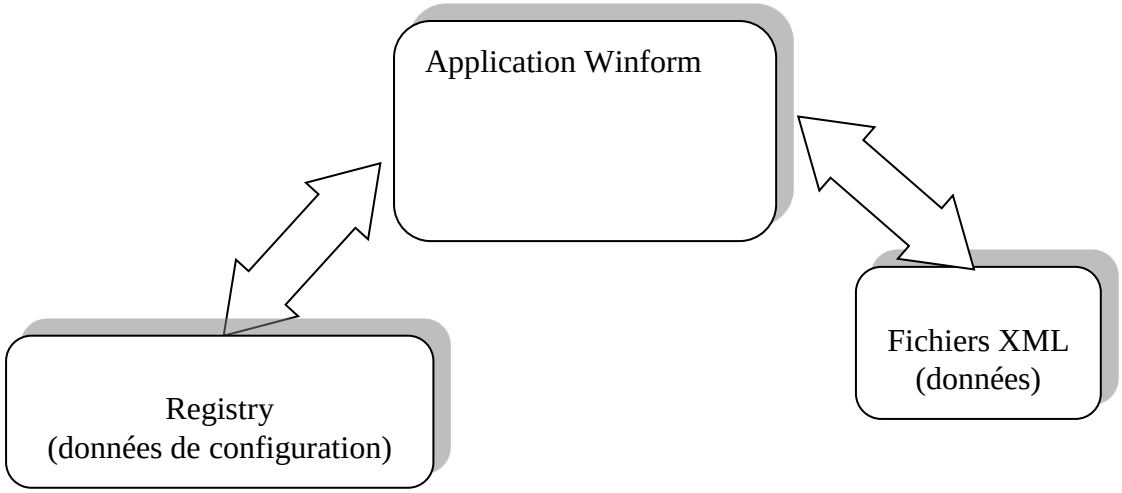
# 3. Phase 3 : Application finale

## *Mise en situation et objectifs*

La société *InpresSoftwareTeam* s'est engagée à fournir un logiciel aux organismes régionaux responsables de l'octroi de primes environnementales aux entreprises et aux particuliers.
L'application "Optimize Energy consumption" permet de déterminer la consommation d'énergie moyenne d'une maison ou d'un immeuble quelconque. En fonction des résultats obtenus l'organisme décidera d'octroyer ou non la prime « HomeEnergySurvey ».
L'application ne prend pas la décision. Elle la suggère à un utilisateur qui a le loisir/la responsabilité de l'appliquer ou non en fonction de paramètres supplémentaires.
L'objectif du logiciel est de simuler la dépense énergétique moyenne d'une habitation. Dans une première version, l'application est mise à la disposition de deux profils d'utilisateurs :

- Les propriétaires d'habitations
- Les administrateurs de l'application, membre du personnel de la société *InpresSoftwareTeam*

## *Architecture*



## *Les composants de l'application*

### 1.1.1 Les utilisateurs

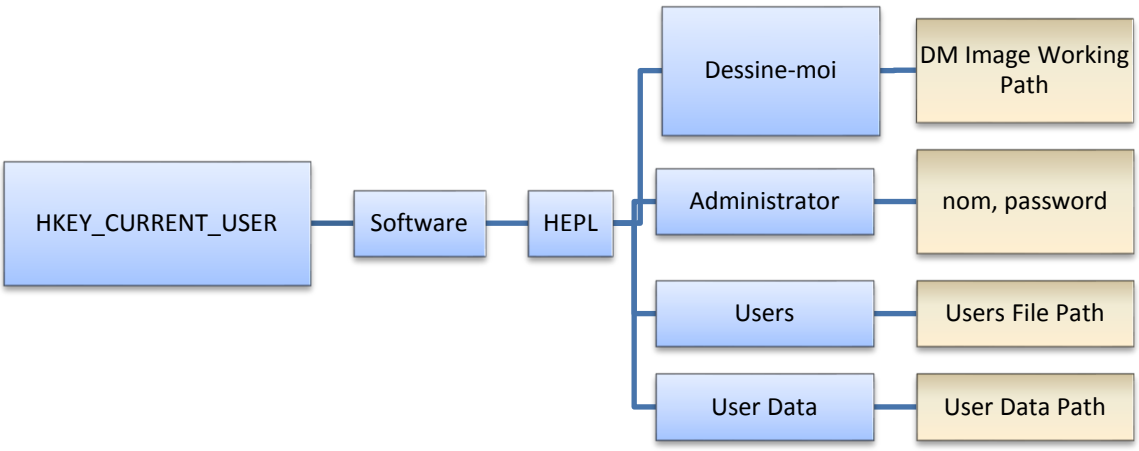
Les utilisateurs doivent s'authentifier pour pouvoir accéder aux différentes fonctionnalités de l'application. Les caractéristiques des comptes des utilisateurs sont

- Un nom
- Un mot de passe
- Un profil choisi parmi deux types définis :
  - propriétaire
  - administrateur

Les données de l'administrateur : login et mot de passe sont enregistrées dans la registry (voir classe Registry : http://msdn.microsoft.com/fr-fr/library/microsoft.win32.registry%28v=vs.110%29.aspx ). L'administrateur par défaut est "Admin" (sans mot de passe). Dès que l'administrateur modifie son mot de passe, il est enregistré dans la registry.
L'ensemble des comptes utilisateurs est stocké dans un fichier XML (voir classe XmlSerializer : http://msdn.microsoft.com/fr-fr/library/system.xml.serialization.xmlserializer%28v=vs.110%29.aspx ).
Les fonctionnalités disponibles pour les utilisateurs dépendent de leur profil.

### 1.1.2 Données de la registry

La registry permet maintenant de mémoriser plusieurs informations.




Elle contient :

- le nom et le mot de passe de l'administrateur.
- Le nom du dossier qui contient le fichier des utilisateurs (propriétaires de maisons).
- Le nom du dossier qui contient les dossiers des maisons des utilisateurs.

### 1.1.3 Les composants électriques

Nos habitations sont inondées d'appareils en tout genre qui consomment de l'électricité. Parmi ceux-ci, on trouvera :

- les éclairages,
- les appareils électroménagers
- le chauffage,
- la TV, la chaîne hifi, l'ordinateur,
- les appareils autonomes et mobiles qu'il faut charger régulièrement (GSM, appareil photo)

Voyant cette liste, il est assez aisé d'identifier un certain nombre de caractéristiques :

- le type (éclairage, électroménager, loisir, ...)
- la marque
- le modèle
- le mode de fonctionnement (avec ou sans veille (BONUS))
- des données (avec ou sans veille (BONUS)) qui permettront à terme de calculer l'énergie consommée et donc d'évaluer les coûts de consommation.
- Les moments pendants lesquels l'appareil est utilisé sur une période de 24h.
- *un nom d'image représentant l'appareil.*
- ...

Les composants électriques créés par un propriétaire sont stockés dans un fichier XML situé dans le dossier du propriétaire.

### 1.1.4 Les maisons

Les maisons sont composées de plusieurs niveaux (étages, rez-de-chaussée, cave). Les niveaux sont composés de pièces (chambre, cuisine, living, bureau, garage) qui seront représentée graphiquement dans la fenêtre.
Chaque maison est identifiée par un ID unique. Sa description (niveaux, pièces et contenu) ainsi que son adresse sont mémorisées dans un fichier XML situé dans le dossier du propriétaire.
L'interface utilisateur permet d'attribuer des composants électriques aux pièces de la maison. Cela vaut également pour les composants "mobiles".

### 1.1.5 Les calculs

L'application permet d'effectuer différents calculs de consommation pour une période donnée de 1 jour.


*Calcul de la consommation pour une période d'une semaine, un mois, une année*

Les résultats des calculs peuvent être affichés pour :

- Un appareil
- Tous les appareils d'une pièce
- Tous les appareils d'un niveau
- Tous les appareils d'une maison

### 1.1.6 Les formules

Au-delà des calculs de l'énergie consommée, vous utiliserez une facture d'électricité récente pour calculer les coûts en euros. La simulation sera réalisée sur base d'un tarif unique.
Exemples de calcul :

1. Une ampoule de 50W allumée pendant 6h par jour. Energie consommée : 50W * 6h = 300Wh par jour.
2. Un fer à repasser 2200 W allumé pendant 5h par semaine. Energie consommée : 2200W * 5h = 11000 Wh = 11 KWh par semaine.
3. Un ordinateur portable utilise 4,74A à 19V pendant 8h par jour. Energie consommée : 4,74A * 19V * 8h = 720Wh par jour.

Nous espérons obtenir des résultats plausibles issus de vos calculs.

### 1.1.7 Le simulateur

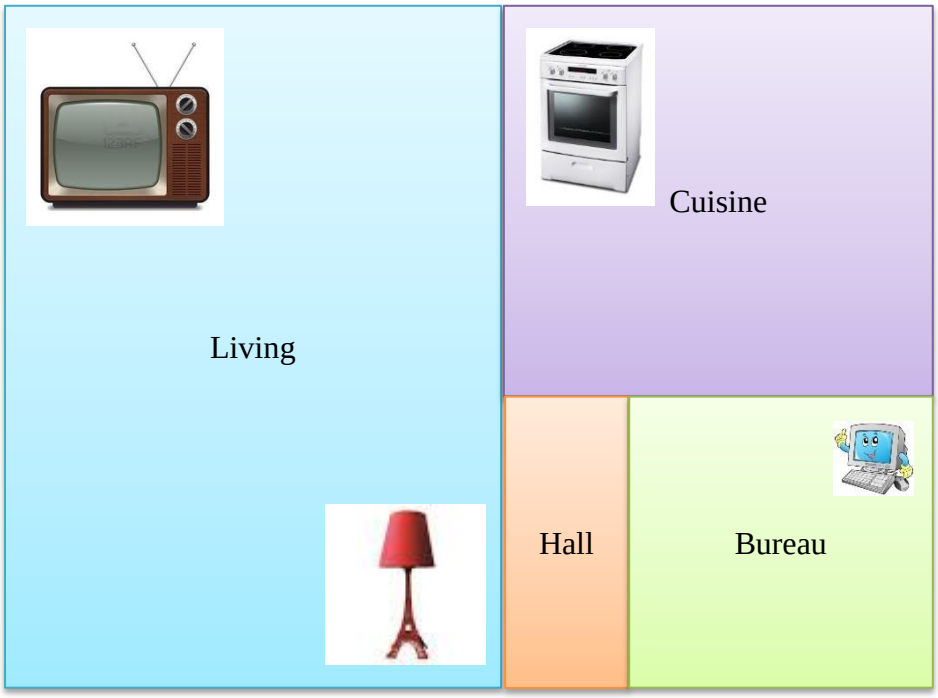
A l'aide d'un timer qui parcours 24h par tranche de 1/2h, montrer les moments pendant lesquels les appareils du niveau en cours de visualisation sont allumés. Suggestion : Les encadrer de couleurs vive lorsqu'ils sont allumés.

## *Fonctionnalités*

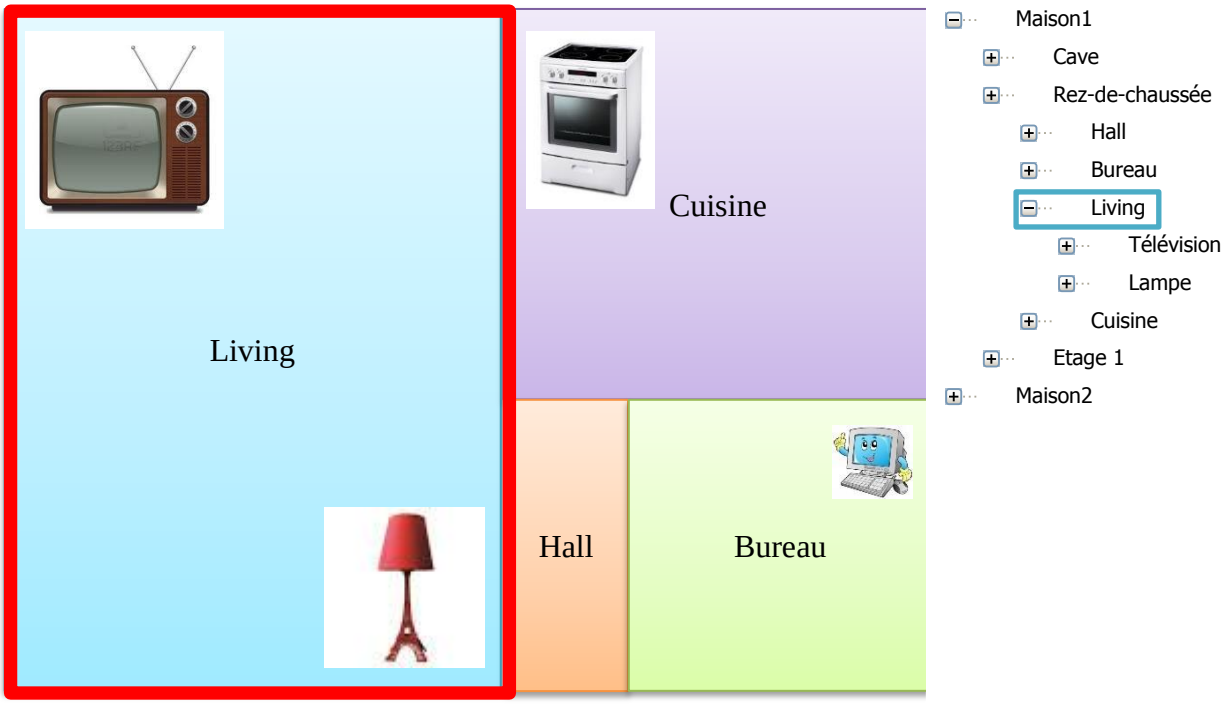
Les fonctionnalités à développer seront accessibles aux utilisateurs en fonction de leur profil.

| Fonctionnalités | Propriétaire | Administrateur |
|---|---|---|
| 1. Ajout / Modification / Suppression d'appareils : | X | |
| 1.1. Encodage des caractéristiques de ceux-ci | X | |
| 1.2. Choix d'image | X | |
| 1.3. Enregistrement du fichier contenant les appareils (fichier XML) | X | |
| 2. Création d'une maison, définition du nombre de niveaux ainsi que des pièces faisant partie des niveaux. | X | |
| 3. Visualisation de la structure de la maison dans une TreeView. Un propriétaire y voit toutes ses maisons. | X | |
| 3.1. La sélection de la maison provoque l'affichage de l'adresse de celle-ci ainsi que sa photo. | X | |
| 3.2. La sélection d'un niveau provoque l'affichage du dessin du niveau, donc des différentes pièces qu'il contient ainsi que du contenu de chacune d'elles. Elle permet l'édition des données du niveau. | X | |
| 3.3. La sélection d'une pièce l'encadre en couleur dans le dessin du niveau. Elle affiche les détails de la pièce. | X | |
| 3.4. La sélection d'un composant électrique l'encadre en couleur dans le dessin et affiche les détails de celui-ci. La commande d'ajout d'appareil se fait par click dans la pièce à laquelle il doit être attaché. Il est sélectionné parmi la liste des appareils disponibles. Plusieurs appareils du même type peuvent apparaître dans une maison. Il est possible de supprimer un appareil, de le déplacer dans une pièce. | X | |
| 4. Calcul des énergies consommées | X | |
| 5. Ajout / Modification / Suppression d'un utilisateur | | X |
| 6. Modification du nom et du mot de passe administrateur | | X |

### 1.1.8 Exemple de visualisation de niveau




### 1.1.9 Exemple de visualisation d'une pièce (le living)



### 1.1.10 Récapitulatif des attentes minimales au niveau « POO – C# .NET »

Enfin, il va de soi que la solution proposée devra présenter une interface graphique attrayante, intuitive et pratique pour l'utilisateur (ex : copier-coller pour dupliquer la configuration d'une habitation ou drag and drop pour DEPLACER un élément d'une pièce à une autre, etc …). Elle devra se mouler dans la mouvance des interfaces graphiques des applications Windows actuelles. La solution devra être codée de manière orientée objets (Des classes, des Interfaces, des événements, des listes de …, etc.).

## *A définir avant de commencer à développer*

A montrer pour validation au prof de laboratoire avant le 4 avril 2014.

1. Liste des objets à mettre en œuvre, propriétés et méthodes, diagramme de classes
2. Nom des fichiers XML, définition des balises à mettre en œuvre
3. Définition de l'interface utilisateur
4. Logique d'utilisation de l'interface utilisateur
5. ....

# 4. Planning

Les dates ci-dessous font référence au premier jour de la semaine (celle-ci sont évidemment à adapter en fonction du jour de la séance de laboratoire).

| | | |
|---|---|---|
| Semaine 1 | 20/01/2014 | Phase 1 |
| Semaine 2 | 27/01/2014 | Phase 1 |
| Semaine 3 | 03/02/2014 | Phase 1 |
| Semaine 4 | 10/02/2014 | *Phase 2* |
| Semaine 5 | 17/02/2014 | **Phase 1 EVALUATION** |
| Semaine 6 | 24/02/2014 | Phase 2 |
| | 03/03/2014 | *Congé* |
| Semaine 7 | 10/03/2014 | Phase 2 |
| Semaine 8 | 17/03/2014 | Phase 3 & **Phase 2 EVALUATION** |
| Semaine 9 | 24/03/2014 | Phase 3 |
| Semaine 10 | 31/03/2014 | Phase 3 |
| | 07/04/2014 | *Congé* |
| | 14/04/2014 | *Congé* |
| Semaine 11 | 21/04/2014 | Phase 3 |
| Semaine 12 | 28/05/2014 | Phase 3 |
| Semaine 13 | 05/05/2014 | Phase 3 |
| Semaine 14 | 12/05/2014 | Phase 3 |

# 5. Contenu du dossier final

Le dossier final permet de valoriser votre travail. Il est imprimé et envoyé par mail dans l'archive qui contient également toutes les sources du programme.

Il devra contenir les éléments suivants :

1. Bref rappel de l'énoncé,
2. Liste des fonctionnalités développées, y compris les extra,
3. Les choix technologiques,
4. Une description :
    a. des classes,
    b. des listes templates,
    c. des événements spécialement ajoutés (présentés dans leur contexte),
    d. de la source de donnée principale ainsi que des différents binding sources intégrés dans les différents User Control (schéma et description)
5. Un manuel d'utilisation de l'application, y compris des copies d'écran de l'interface utilisateur ainsi que des commentaires sur celle-ci,
6. Une conclusion.

# 6. Règles d'évaluation

**1)** L'évaluation établissant la note du cours d'application programmation orientée objet est réalisée de la manière suivante :

♦ théorie : un examen écrit en juin 2014 (sur base d'une liste de questions disponibles sur l'école virtuelle et à préparer);

♦ laboratoire : le laboratoire sera composé de trois parties distinctes. Les deux premières sont soumises à une évaluation continue (aux dates précisées dans l'énoncé de laboratoire), la troisième est évaluée lors de l'examen oral de juin 2014. La cote finale de laboratoire sera la moyenne pondérée des cotes obtenues dans chacune des trois parties (10%, 30% et 60%).

♦ note finale : **moyenne de la note de théorie (50%) et de la note de laboratoire (50%)**.

Cette procédure est d'application tant en 1ère qu'en 2ème session, ainsi que lors d'une éventuelle prolongation de session.

**2)** En 2ème session, un **report de note** est possible pour chacune des deux parties de laboratoire ainsi que pour la note de théorie **pour des notes supérieures ou égales à 10/20**.
Toutes les évaluations (théorie ou laboratoire) ayant des **notes inférieures à 10/20** sont **à représenter dans leur intégralité**.

**3)** Les consignes de présentation des dossiers de laboratoire sont stipulées dans l'énoncé de laboratoire disponible dans le centre de ressources.