C#, UML, MVC, *Visual Studio* Cheat Sheet

1 Classe

1.1 Définition

```
using System;
using ...

namespace Name {
  class ClassName : SuperClass {
    // fields

    // properties (getter/setter)

    // constructors

    // methods
  }
}
```

2 Convention de noms

2.1 Classe

Nom débutant par une majuscule.

2.2 Objet

Nom débutant par une minuscule.

2.3 Champs, propriétés

Champ (privé) : nom commun débutant par le caractère « souligné » suivi d'une minuscule (*CamelCase*).
Propriété (publique) : nom commun débutant par une majuscule (*PascalCase*).

```
private int _numerator;
public int Numerator {
  get { return _numerator; }
  set { _numerator = value; }
}
```

2.4 Constructeur

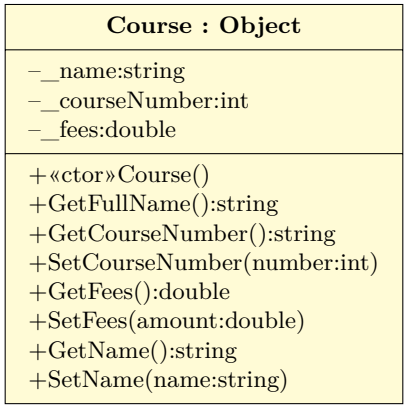
Nom identique à celui de la classe.

2.5 Méthode

Verbe d'action débutant par une majuscule.

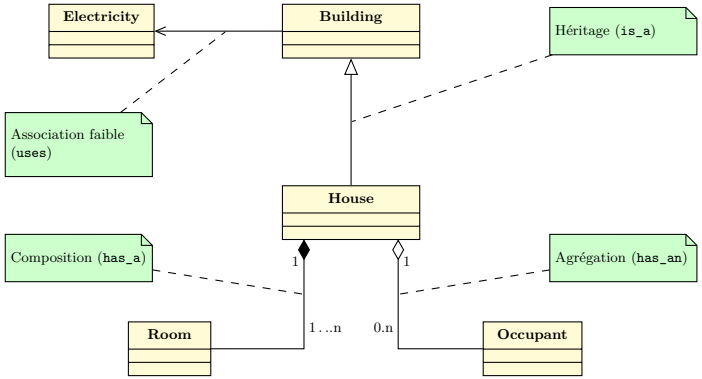
3 UML

3.1 Diagramme de classe



– : champ/méthode privé, **+** : champ/méthode public

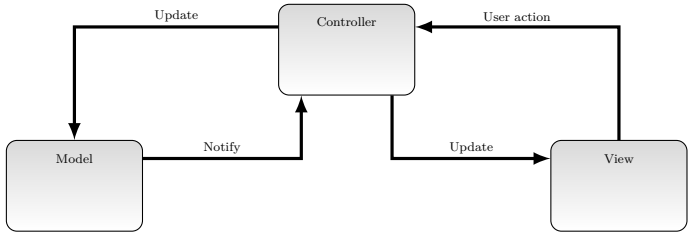
3.2 Liaisons entre classes



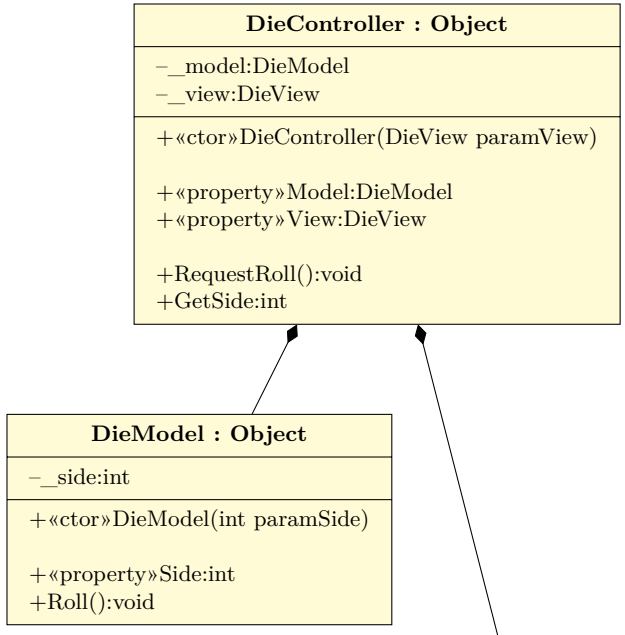
Liaison	Link	*Symbole*	*Exemple*
association faible	*weak association*	⟶	A House `USES` Electricity
composition	*composition*	⟶◆	A House `HAS_A` Room.
agrégation	*agregation*	⟶◇	A House `HAS_AN` Occupant.
héritage	*inheritance*	⟶▷	A House `IS_A` Building.

4 MVC (*Model-View-Controller*)

4.1 Principe



4.2 Diagramme de classes



5 Exemple MVC minimal : *Die* (dé)

Rem. : projet *Visual Studio* attaché au PDF.

```
[System.Serializable]
public class DieModel : Object {
  static private int SIDE_MIN = 1, SIDE_MAX = 6;
  private int _side;
  private Random _rnd;

  // getters/setters
  public int Side {
    get { return _side; }
    private set { _side = value; }
  }
  private Random Rnd {
    get { return _rnd; } set { _rnd = value; }
  }

  // constructors
  public DieModel(): this(SIDE_MIN) {}
  // designated constructor
  public DieModel(int paramSide) {
    this.Side = paramSide; this.Rnd = new ↩
        Random();
  }

  // methods
  public void Roll(){
    this.Side = this.Rnd.Next(SIDE_MIN, ↩
        SIDE_MAX+1);
  }
}//
```

Listing 1 – `DieModel.cs`

```
class DieController : Object {
  private DieModel _model;
  private DieView _view;

  // getters/setters
  private DieModel Model {
    get { return _model; } set { _model = ↩
        value; }
  }
  private DieView View {
    get { return _view; } set { _view = value; }
  }

  // constructors
  public DieController(DieView paramView) {
    this.View = paramView;
    this.Model = new DieModel(1);
  }

  // methods
  public void RequestRoll() {
    this.Model.Roll();
    this.View.UpdateView();
  }

  public double GetSide() { return ↩
      this.Model.Side; }
}//
```

Listing 2 – `DieController.cs`

```
public partial class DieView : Form {
  private DieController _controller;

  // getters/setters
  private DieController Controller {
    get { return _controller; } set { ↩
        _controller = value; }
  }

  // constructors
  public DieView() {
    InitializeComponent();
    this.Controller = new DieController(this);
  }

  // methods
  private void DieView_Load(object sender, ↩
      EventArgs e) {
    this.UpdateView();
  }

  private void btnRoll_Click(object sender, ↩
      EventArgs e) {
    this.Controller.RequestRoll();
  }

  public void UpdateView() {
    tebSide.Text = ↩
        this.Controller.GetSide().ToString();
  }
}//
```

Listing 3 – `DieView.cs`

6 Sérialisation/déserialisation

```
public class DieController : Object {
  private DieModel _model;
  ...
  public void SaveData() {
    FileStream fs = new FileStream(FILENAME, ↩
        FileMode.Create);

    XmlSerializer xs = new ↩
        XmlSerializer(typeof(DieModel));
    xs.Serialize(fs, this.Model);

    fs.Close();
  }

  public void LoadData() {
    if (File.Exists(FILENAME)) {
      FileStream fs = new FileStream(FILENAME, ↩
          FileMode.Open);

      XmlSerializer xs = new ↩
          XmlSerializer(typeof(DieModel));
      this.Model = (DiceModel)xs.Deserialize(fs);

      fs.Close();
    }
}}
```

7 Raccourcis *Visual Studio*

7.1 Génération/exécution/débogage

F5	démarre le débogage
SHIFT+F5	arrête le débogage
CTRL+F5	exécute sans débogage
F6	génère la solution
F10/F11	exécute un pas (principal/détaillé)
CTRL+R, A	exécute tous les tests de la solution
CTRL+R, T	exécute les tests du contexte actuel
CTRL+W, O	fenêtre des messages *output* `Trace.Write()`

7.2 Complétion

TAB	insère un modèle de code (`class`, `ctor`, `cw`, `if`, `for`, `forr`, `foreach`, `prop`, `propg`, `switch`, `while`.) 2 × TAB pour un modèle pré-rempli
CTRL+SPACE	compléte le mot (variable, méthode.)

7.3 Formatage

CTRL+K, CTRL+F	auto-indente la sélection
CTRL+K, CTRL+D	auto-indente le fichier
CTRL+K, CTRL+S	entoure la sélection avec un *snippet* (`#if`, `#region`, `for`, `foreach`.)
CTRL+K, CTRL+C	commente un bloc
CTRL+K, CTRL+U	décommente un bloc
CTRL+K, CTRL+X	insère un *snippet*

7.4 Navigation/fenêtre

F7	affiche la fenêtre de code
F12	atteindre la définition d'un élément
SHIFT+F7	affiche la fenêtre d'interface graphique
CTRL+K, R	recherche toutes les références
CTRL+W, D	affiche la fenêtre de définition de code (classe, structure.)
CTRL+TAB	affiche la liste des fichiers source ouverts
CTRL+,	affiche la fenêtre de navigation (recherche dans le projet)

7.5 Génération classe, `getter/setter`

SHIFT+ALT+C	ajoute un fichier de classe
CTRL+R, E	encapsule le champ

7.6 Plan

CTRL+M, M	plie/déplie une région (`#region ... #endregion`)
CTRL+M, L	plie/déplie toutes les régions

Csharp_UML_MVC_cheat_sheet_chm.tex, V5672, 28/08/2014, C. Maréchal
Auteur : C. Maréchal, CFPT - École d'Informatique.