Design patterns

Design patterns

- Définition modèles de conception.
- Patrons de conception du GoF (*Gang of Four - GoF*).
- Utilisation modèles de conception.
- Avantages et inconvénients
- Exemple d’un modèles de conception.

Types Design patterns

- **Design Patterns - Factory Pattern**
- Abstract Factory Pattern
- **Design Patterns - Singleton Pattern**
- **Design Patterns - Builder Pattern**
- Design Patterns - Prototype Pattern
- Design Patterns - Adapter Pattern
- Design Patterns - Bridge Pattern
- Design Patterns - Filter Pattern
- **Design Patterns - Composite Pattern**
- Design Patterns - Decorator Pattern
- Design Patterns - Facade Pattern
- Design Patterns - Flyweight Pattern
- Design Patterns - Proxy Pattern
- Chain of Responsibility Pattern

Types Design patterns

- **Design Patterns - Command Pattern**
- Design Patterns - Interpreter Pattern
- Design Patterns - Iterator Pattern
- Design Patterns - Mediator Pattern
- Design Patterns - Memento Pattern
- **Design Patterns - Observer Pattern**
- Design Patterns - State Pattern
- Design Patterns - Null Object Pattern
- Design Patterns - Strategy Pattern
- Design Patterns - Template Pattern

Types Design patterns

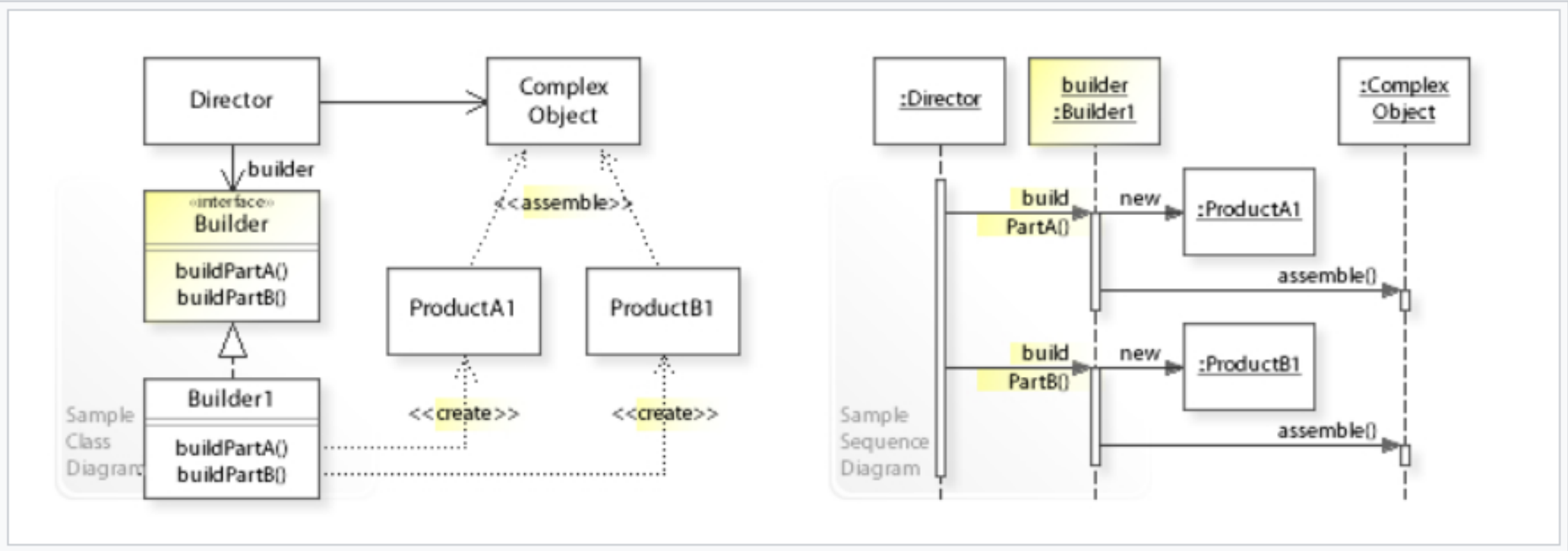
- Design Patterns - Visitor Pattern
- Design Patterns - MVC Pattern
- Business Delegate Pattern
- Composite Entity Pattern
- Data Access Object Pattern
- Front Controller Pattern
- Intercepting Filter Pattern
- Service Locator Pattern
- Transfer Object Pattern

Design patterns: Builder

- Définition: Séparer la construction d'un objet complexe de sa représentation afin que le même processus de construction puisse créer des représentations différentes.

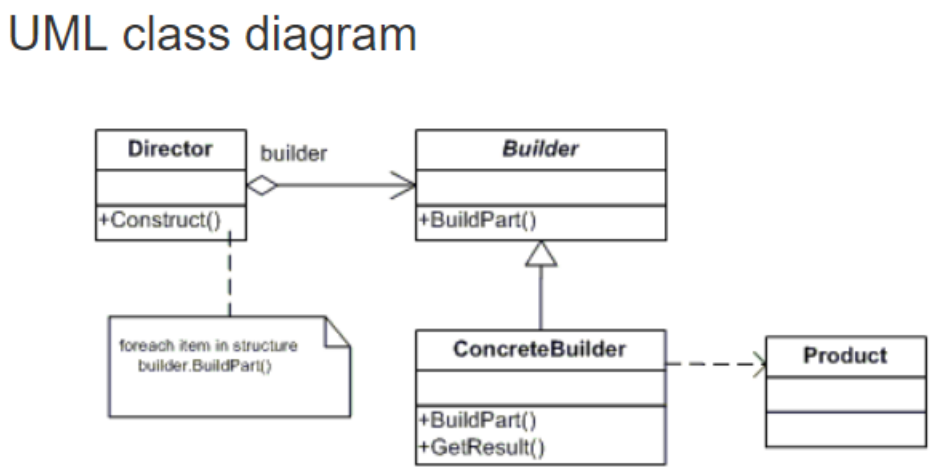
Design patterns: Builder

UML class and sequence diagram



Design patterns: Builder

- UML class diagramme



Design patterns: Builder

Les avantages du modèle Builder.

- Vous permet de faire varier la représentation interne d'un produit.
- Encapsule le code pour la construction et la représentation.
- Fournit un contrôle sur les étapes du processus de construction.

Design patterns: Builder

Les inconvénients du modèle Builder,

- Nécessite la création d'un **ConcreteBuilder** distinct pour chaque type de produit.
- Requiert que les classes **Builder** soient modifiables.
- Les membres de classe de données ne sont pas garantis d'être initialisés.
- L'injection de dépendance peut être moins soutenue

Design patterns: Builder

Participants

- Les classes et objets participant à ce modèle sont:
- **Constructeur** (VehicleBuilder)
- spécifie une interface abstraite pour créer des parties d'un objet Product
- **ConcreteBuilder** (MotorCycleBuilder, CarBuilder, ScooterBuilder)
- construit et assemble des parties du produit en implémentant l'interface Builder
- définit et suit la représentation qu'il crée
- fournit une interface pour récupérer le produit
- **Directeur** (Boutique)
- construit un objet en utilisant l'interface Builder
- **Produit** (véhicule)
- représente l'objet complexe en construction. ConcreteBuilder construit la représentation interne du produit et définit le processus par lequel il est assemblé
- inclut des classes qui définissent les parties constituantes, y compris des interfaces pour assembler les parties dans le résultat final

Design patterns: Builder

Exemple: C#

```
/// <summary>
/// Represents a product created by the builder
/// </summary>
public class Car
{
    public string Make { get; }
    public string Model { get; }
    public int NumDoors { get; }
    public string Colour { get; }

    public Car(string make, string model, string colour, int numDoors)
    {
        Make = make;
        Model = model;
        Colour = colour;
        NumDoors = numDoors;
    }
}
```

Design patterns: Builder

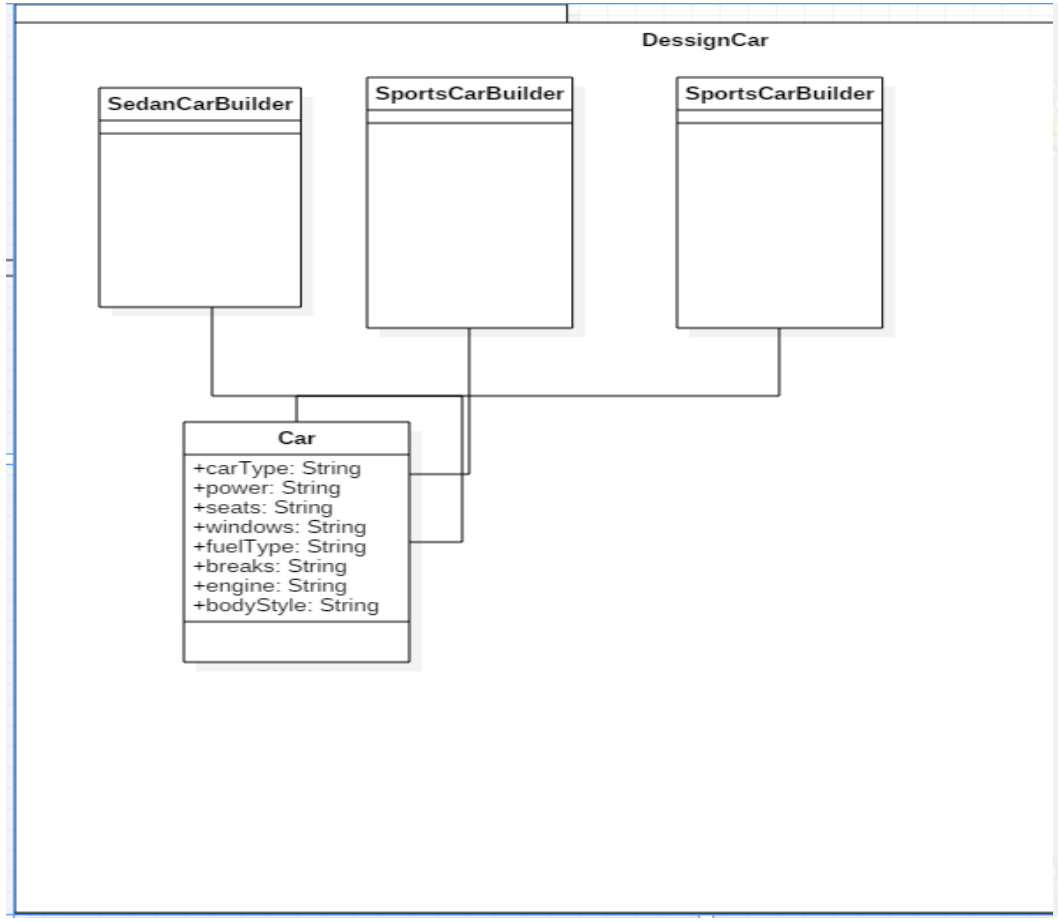
```
/// <summary>
/// The builder abstraction
/// </summary>
public interface ICarBuilder
{
    string Colour { get; set; }
    int NumDoors { get; set; }

    Car GetResult();
}

/// <summary>
/// Concrete builder implementation
/// </summary>
public class FerrariBuilder : ICarBuilder
{
    public string Colour { get; set; }
    public int NumDoors { get; set; }

    public Car GetResult()
    {
        return NumDoors == 2 ? new Car("Ferrari", "488 Spider", Colour, NumDoors) : null;
    }
}
```

Design patterns: TP Builder

- Le but de ce TP est :de voir les design patterns : méthode builder,
- On souhaite créer une application que permet afficher les détail technique d’un voiture: (Sport, Belin, sedan etc..) en utilisant le design pattern Builder.

Design patterns: TP Builder

Diagramme classe




```
public interface CarBuilder {

        public void buildBodyStyle();
        public void buildPower();
        public void buildEngine();
        public void buildBreaks();
        public void buildSeats();
        public void buildWindows();
        public void buildFuelType();
        public Car getCar();
}
```

Design patterns: TP Builder

- L'interface de CarBuilder est l'outil de construction contient un ensemble de méthodes communes utilisées pour construire l'objet voiture et ses composants

- La méthode getCar est utilisée pour renvoyer l'objet Car final au client après sa construction. Voyons deux implémentations de l'interface CarBuilder, une pour chaque type de voiture, c'est-à-dire pour une berline et une voiture de sport

Design patterns: TP Builder

Affichage:

```
--------------SEDAN--------------------
Body: External dimensions: overall length (inches): 202.9, overall width (inches): 76.2,  ↩
    overall height (inches): 60.7, wheelbase (inches): 112.9, front track (inches): 65.3,  ↩
    rear track (inches): 65.5 and curb to curb turning circle (feet): 39.5
Power: 285 hp @ 6,500 rpm; 253 ft lb of torque @ 4,000 rpm
Engine: 3.5L Duramax V 6 DOHC
Breaks: Four-wheel disc brakes: two ventilated. Electronic brake distribution
Seats: Front seat center armrest.Rear seat center armrest.Split-folding rear seats
Windows: Laminated side windows.Fixed rear window with defroster
Fuel Type: Gasoline 19 MPG city, 29 MPG highway, 23 MPG combined and 437 mi. range
--------------SPORTS--------------------
Body: External dimensions: overall length (inches): 192.3, overall width (inches): 75.5,  ↩
    overall height (inches): 54.2, wheelbase (inches): 112.3, front track (inches): 63.7,  ↩
    rear track (inches): 64.1 and curb to curb turning circle (feet): 37.7
Power: 323 hp @ 6,800 rpm; 278 ft lb of torque @ 4,800 rpm
Engine: 3.6L V 6 DOHC and variable valve timing
Breaks: Four-wheel disc brakes: two ventilated. Electronic brake distribution. StabiliTrak ↩
     stability control
Seats: Driver sports front seat with one power adjustments manual height, front passenger  ↩
    seat sports front seat with one power adjustments
Windows: Front windows with one-touch on two windows
Fuel Type: Gasoline 17 MPG city, 28 MPG highway, 20 MPG combined and 380 mi. range
```