2. Будівельник (Builder)

Уявіть, що ви володієте магазином (гаражем) з продажу персональних комп'ютерів, в якому можна вибирати конфігурацію прямо біля каси (як піцу в піцерії). Вам слід створити систему, що дозволить легко *будувати* будь-яку конфігурацію ноутбука для будь-якого покупця. Причому, стандартні конфігурації мають складатися «по накату». От як піца може бути «фірмова», «грибна», «справжня італійська», «пепероні», або ваша специфічна, так і ноутбук: для геймера або для подорожей. Влаштуємо бізнес?

Будівельник вимальовує стандартний процес створення складного об'єкта, розділяючи логіку будування об'єкта від його представлення.[14]

Шляхом додавання певних частин, таких як процесор, пам'ять, жорсткий диск, батарея і т.д, можна скласти повноцінний комп'ютер. Часто конфігурації можуть набути певних стандартів. В кінці кінців, ваш продавець балакає із клієнтом і запитує, яку пам'ять він хоче, і так далі. Або ж, якщо клієнт не дуже «шарющий», продавець може запитати: «Ну вам той комп треба для шпільок? Яких?». Звісно, що ви наперед знаєте певні кроки, щоб створити ноутбук. Ці кроки визначаються в *абстрактному будівельнику* (*abstract builder*). Єдине що вам потрібно від клієнта, це дізнатися, які компоненти використовувати на кожному кроці.

Уривок коду 2.1. Абстракний будівельник

```
abstract class LaptopBuilder
{
    protected Laptop Laptop { get; private set; }
    public void CreateNewLaptop()
    {
        Laptop = new Laptop();
    }
    // Метод, який повертає готовий ноутбук назовні
    public Laptop GetMyLaptop()
    {
        return Laptop;
    }
    // Кроки, необхідні щоб створити ноутбук
    public abstract void SetMonitorResolution();
    public abstract void SetProcessor();
    public abstract void SetMemory();
    public abstract void SetHDD();
    public abstract void SetBattery();
}
```

[14] **Builder**. Intent. Separate the construction of a complex object from its representation so that the same construction process can create different representations.
Будівельник. Призначення. Розділити створення складного об'єкта від його представлення, щоб той же процес створення міг утворити різні представлення.

Якщо покупець відповідає «Ух, так! Я хочу шпіляти... yyeex!» і у нього загорілися очі, то ви вже напоготові і маєте конкретну реалізацію для ігрового ноутбука:

Уривок коду 2.2. Конкретна реалізація будівельника

```
// Таким будівельником може бути працівник, що
// спеціалізується у складанні «геймерських» ноутів
class GamingLaptopBuilder : LaptopBuilder
{
    public override void SetMonitorResolution()
    {
        Laptop.MonitorResolution = "1900X1200";
    }
    public override void SetProcessor()
    {
        Laptop.Processor = "Core 2 Duo, 3.2 GHz";
    }
    public override void SetMemory()
    {
        Laptop.Memory = "6144 Mb";
    }
    public override void SetHDD()
    {
        Laptop.HDD = "500 Gb";
    }
    public override void SetBattery()
    {
        Laptop.Battery = "6 lbs";
    }
}
```

Але, якщо ваш покупець бізнесмен, який переглядає презентації і звіти, перелітаючи через атлантику:

Уривок коду 2.3. Конкретний будівельник для ноутбуків

```
// А ось інший «збирач» ноутів
class TripLaptopBuilder : LaptopBuilder
{
    public override void SetMonitorResolution()
    {
        Laptop.MonitorResolution = "1200X800";
    }
    public override void SetProcessor()
    {
        //.. і так далі...
```

Для того, щоб справитися із побудовою ноутбука, базуючись на відповіді покупця, знадобиться директор (*director*):

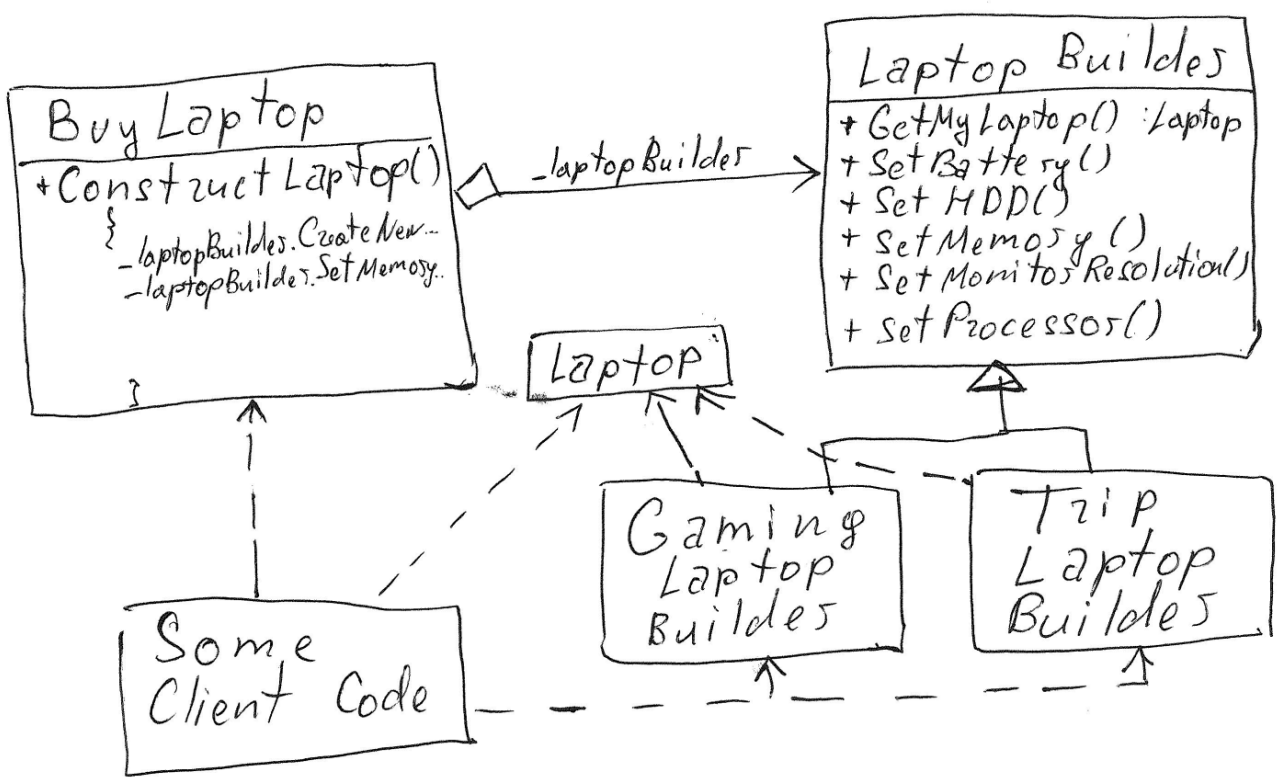
Уривок коду 2.4. Директором може бути код приймаючий замовлення

```
class BuyLaptop
{
    private LaptopBuilder _laptopBuilder;
    public void SetLaptopBuilder(LaptopBuilder lBuilder)
    {
        _laptopBuilder = lBuilder;
    }
```

```
        // Змушує будівельника повернути цілий ноутбук
        public Laptop GetLaptop()
        {
            return _laptopBuilder.GetMyLaptop();
        }
        // Змушує будівельника додавати деталі
        public void ConstructLaptop()
        {
            _laptopBuilder.CreateNewLaptop();
            _laptopBuilder.SetMonitorResolution();
            _laptopBuilder.SetProcessor();
            _laptopBuilder.SetMemory();
            _laptopBuilder.SetHDD();
            _laptopBuilder.SetBattery();
        }
    }
```

Уривок коду 2.5. А тепер глянемо на використання цього патерну

```
// Ваша система може мати багато конкретних будівельників
var tripBuilder = new TripLaptopBuilder();
var gamingBuilder = new GamingLaptopBuilder();
var shopForYou = new BuyLaptop();//Директор
// Покупець каже, що хоче грати ігри
shopForYou.SetLaptopBuilder(gamingBuilder);
shopForYou.ConstructLaptop();
// Ну то нехай бере що хоче!
Laptop laptop = shopForYou.GetLaptop();
Console.WriteLine(laptop.ToString());
// Вивід: [Laptop: 1900X1200, Core 2 Duo, 3.2 GHz, 6144 Mb, 500 Gb, 6 lbs]
```



UML-діаграма 2. Будівельник